أوراكل للبرمجة مع ڤيچوال بيسك

Oracle Programming with Visual Basic

تأليف

نيك ستــودون

أوراكل للبرمجة مع
فيچوال بيسك

Oracle Programming
with Visual Basic

دار الفاروق للنشر والتوزيع

أكبر مركز في الشرق الأوسط
لإصدار أحدث الكتب في عالم الكمبيوتر

العنوان

فرع وسط البلد: ٣ شارع منصور - المبتديان - متفرع من شارع مجلس الشعب محطة مترو سعد زغلول - القاهرة - مصر.

تليفون : ٣٥٥٣٠٣٢ (٠٠٢٠٢) - ٣٥٤٣٢٠٣ (٠٠٢٠٢)

فاكس : ٣٥٤٣٦٤٣ (٠٠٢٠٢)

فرع الدقي: ١٢ شارع الدقي الدور السابع - إتجاه الجامعة مَنْزَل كوبري الدقي

تليفون : ٣٣٨١٠٢٢ (٠٠٢٠٢)

فاكس : ٣٣٨٢٠٧٤ (٠٠٢٠٢)

الطبعة العربية الأولى (يناير ٢٠٠٠)

عدد الصفحات: ٧٣٦

رقم الإيداع: ١٣٦١٤ لسنة ١٩٩٩

الترقيم الدولي: 977-307-061-1

إعداد وترجمة

دار الفاروق وجي. إن. إس

لمحة على المحتويات

المحتويات

٢ بنية قاعدة البيانات لأوراكل ٥٧

٤ إدارة قاعدة البيانات ١٢٩

٦ تنفيذ قاعدة البيانات ١٩٣

٨ تحديث قاعدة البيانات ٢٨٧

٩ مقدمة عن PL/SQL ٣١١

١٠ تكوين الإجراءات المخزنة ٣٤٥

١٢ الوصول لأوراكل من خلال فيجوال بيسك ٤٣٣

الجزء الخامس: أساليب متقدمة لأوراكل ٥٤١

١٥ وحدات البيانات البعيدة ٥٤٣

١٧ وحدات بيانات ActiveX ٦١٧

المقدمة

ما الحاجة لكتاب عن أوراكل وفيجوال بيسك

هذا الكتاب متميز في مجال الكمبيوتر إذ أنه يغطي موضوعين مختلفين ومنفصلين من موضوعات تقنية المعلومات. توجد العديد من الكتب الشاملة والجيدة عن أوراكل. وهي مصدر مصمم لتوضيح المعلومات لمسئولي قواعد البيانات. ويوجد عدد أكبر من الكتب عن فيجوال بيسك ولكن لن تجد في هذه الكتب أي إشارة لأوراكل ولا حتى في الفهرست. وفي الأعوام القليلة الماضية واجهتي مشكلة نقص المعلومات، حتى مع معرفتي الجيدة بفيجوال بيسك "الإصدار ٣ في هذا الوقت"، وجدت بفاعلية ومقدرة إنني أمام مشكلة تحتاج لحل.

ويُعد هذا الكتاب كتاباً شخصياً، لأنه يقدم لك المرجع الذي كنت أود الحصول عليه منذ أعوام قليلة مضت. فهو يجمع بين شرح عن المفاهيم والمعلومات الأساسية عن أوراكل التي بدونها كنت ستجد صعوبة في الاتصال والتعامل مع مسئولي قواعد البيانات وصعوبة في تنفيذ دور مسئول قاعدة البيانات بنفسك إذا صادفت نفس الموقف (قاعدة بيانات أوراكل ليست أبسط قاعدة بيانات يمكن إدارتها ولا يأخذ مدراء المشروعات هذا بعين الاعتبار أو التقدير).

عامةٍ تُعد نصوص أوراكل معقدة بعض الشيء للمستخدمين الجدد، الذين اختاروا بعض النقاط الهامة للتدرب عليها وإتقانها وتركوا بعض المعلومات الأخرى التي رأوا تأجيل الإطلاع عليها لحين توافر المزيد من الوقت ليتدربوا عليها بشكل أفضل. أما كتب فيجوال بيسك، فهي إما بسيطة تغطي مقدمة عن الموضوع بشكل عام، أو تتناول Microsoft SQL Server. لذا تترك العديد من النقاط بدون توضيح كافي لمطوري أوراكل، حتى عند التعامل مع مجرد أمثلة عن كيفية عمل أي خاصية أو سمة.

أدى ندرة الكتب الموجهة للمتطورين الذين يستخدمون أوراكل، وعدم تغطية قاعدة بيانات أوراكل في إطار فيجوال يسك إلى الحاجة لتغطية مثل هذه الموضوعات. ولذا سنقدم لك معلومات تساعدك على الانتقال من فيجوال بيسك إلى أوراكل.

لمن هذا الكتاب

الهدف من هذا الكتاب هو توفير ما نسميه برحلة في عالم أوراكل لمطوري فيجوال بيسك المتمرسين.لن تجد أي تفاصيل عن البرمجة باستخدام فيجوال بيسك ما عدا بعض الموضوعات التي تنطبق على الوصول للبيانات، وهذه تستخدم بشكل بسيط مع Access أو SQL. في الجزئية

المسماة "المصادر" الموجودة في هذه المقدمة، وتوجد إشارة لعدة أماكن للبحث عن المعلومات سواء للأشخاص الجدد أو المتقدمين .يجب وضع النقاط التالية في الاعتبار لمعرفة "لمن هذا الكتاب؟"

- تريد الحصول على نظرة المطور لبرنامج أوراكل.
- عليك توفير مساعدة لمسئولي قاعدة البيانات، أو تحتاج تعليمات موجزة لتنفيذ هذا الدور.
- إذا كنت مطور يستخدم فيجوال بيسك، وتستخدم أوراكل حديثاً.
- إذا كنت مطور يستخدم فيجوال بيسك، بشكل متقدم وتحتاج لأحدث معلومات عن كيفية الحصول على أفضل ما في قاعدة بيانات أوراكل.
- تريد كتاب واحد لاستخدامه كمرجع لأوراكل وVB، بدون الرجوع أو البحث في عدد هائل من أدلة المستخدم أو ملفات التعليمات أو غيرها من المعلومات المباشرة.

أقسام الكتاب

ينقسم هذا الكتاب إلى أربعة أقسام، قسمين عن أوراكل من وجهة نظر فجوال بيسك وقسمين عن فجوال بيسك من خلال أوراكل. ثم إعداد الكتاب بحيث يمكن قراءته من أوله لآخره مروراً بالمفاهيم والتعريفات وصولاً إلى تكوين قاعدة البيانات في الجزء الأول وفي الجزء الثاني تبدأ من المفاهيم إلى أساليب أكثر تقدماً. وعلى الرغم من ذلك، يعتبر كل فصل مستقل نسبياً بحيث تستطيع التنقل فيه كما يقرأ لك.

ينقسم الكتاب إلى الأقسام الآتية:

التعريف بأوراكل: يشرح الخصائص التي تميز قاعدة بيانات أوراكل التي تناسب استخدام المؤسسات عن النظام الذي يعتمد على أساس الملفات كما هو الحال مع Access. سنشرح البنية الداخلية لقاعدة بيانات أوراكل بدءاً من عمليات النظام واستخدامات الذاكرة وحتى ملفات البيانات التي تكون مكان التخزين الدائم لقاعدة البيانات، كما ننتقل إلى كيفية تكوين وصيانة قاعدة بيانات أوراكل. كما سنعرض أسس تصميم قاعدة البيانات بدءاً من نموذج عن المفهوم وحتى التنفيذ الفعلي لقاعدة البيانات.

الاستعلامات والإجراءات المخزنة: أولاً يوجد شرح عن كيفية الوصول إلى البيانات وتحديثها في قاعدة البيانات باستخدام أوامر SQL القياسية. يوجد شرح لهذه الأوامر على مستوى بسيط كما توجد معلومات عن كيفية استخدام الخصائص المتقدمة المتاحة في أوراكل. يتضمن هذا القسم أوامر SQL بالإضافة إلى مقدمة عن لغة البرمجة SQL المعتمدة على الإجراءات، متضمنة كيفية كتابة الإجراءات المخزنة التي يمكن أن تتصل بفيجوال بيسك.

تقنية الوصول للبيانات: تعطي نظرة عامة على المكونات المتعددة المطلوبة للاتصال بين الفيجوال بيسك الموجود على المزود وقاعدة بيانات أوراكل الموجود على الخدم. سنعرض حلول الشبكات لأوراكل بالإضافة إلى الطرق التي يتصل بها فجوال بيسك بطبقة الشبكة يقدم هذا القسم لك أكثر تقنيات الوصول للبيانات المستخدمة من فجوال بيسك وتغطي بعض موضوعات التصميم التي تحتاج لمعرفتها قبل بدء التطوير باستخدام التطبيق.

أساليب متقدمة لأوراكل: تعرض بالتفاصيل خمسة من أكثر أساليب وتقنيات الوصول للبيانات استخداماً وانتشاراً. كما سنعرض نماذج الوحدة وكيفية إعداد رابطة مع قاعدة البيانات وكيفية استعادة مجموعات النتائج من قاعدة البيانات. سنعرض كيفية استخدام مجموعات النتائج القابلة للتحديث وتلك المتاحة للقراءة فقط ثم الانتقال للوصول للإجراءات المخزنة وبعض الإمكانات المتقدمة الأخرى. كما سنعرض كيفية استخدام كل طريقة لمعرفة هيكل قاعدة البيانات.

الموارد

توجد العديد من الموارد التي ستجدها مطلوبة لتواكب التغييرات التي تحدث في عالم التقنيات كما ستواجه مشكلات لن نتعرض لها في هذا الكتاب. فيما يلي قائمة شخصية للمؤلف وبالتأكيد ستجد بعض الموارد الخاصة بك لتضيفها لهذه القائمة.

الدورات التدريبية

توفر لك أوركل عدد كبير من الدورات والبرامج التدريبية في العديد من موضوعات قاعدة بيانات أوراكل، بدءا من مقدمة عن SQL وPL/SQL وحتى خادم أوراكل المتوازي المتقدم. تعد هذه الدورات التدريبية هامة لاستيعاب المنتج بشكل جيد. يمكنك الاتصال بفريق دعم أوراكل الموجود لديك للحصول على معلومات أقرب مركز تعليمي لك

يوجد العديد من الدورات التدريبية متاحة في تقنيات مايكروسوفت التي نعطي الدارس شهادات مؤهلة في التطوير. كما نجد أيضا دورات تدريبية في المراكز التعليمية المحلية وبأسعار ملائمة.

الكتب

في أي مكتبة جيدة لبيع الكتب ستجد معلومات متقدمة عن أوراكل بالنسبة لمن يسعى ليكون مسئول عن قاعدة بيانات والعديد من الكتب عن فيجوال بيسك فيما يلي مجموعة من الكتب التي

يمكنك الرجوع إليها، وستجد أن هذا هو الكتاب الوحيدة من فيجوال بيسك الذي يعالج أوراكل بمثل هذا التفصيل. كما هو الحال في كتب أوراكل التي لا تتعرض لفيجوال بيسك بطريقة هامشية.

- إتقان العمل مع فيجوال بيسك ٦ – ايفانجلوس بيترادسوس (سابكس ١٩٩٨).
- الوصول للبيانات باستخدام فيجوال بيسك – جيفري ماكمانوس (سامز ١٩٩٨).
- دليل مطوري قاعدة البيانات باستخدام فجوال بيسك – روجر جينينجز (سامز ١٩٩٧).
- الدليل السريع لفيجوال بيسك وخادم SQL وليام فواجن (مايكروسوفت برتس ١٩٩٦).
- إتقان العمل مع أوراكل ٨:١ – جاتندر بريم (سايكس ١٩٩٩).
- البرمجة مع أوراكل PL/ SQL – ستيفن فيور ستاين (أوريلي ١٩٩٧).
- أداء أفضل مع أوراكل – مارك جاري وبيتر كوريجان أويللي ١٩٩٦).

مجالات دورية

على مر السنوات، اتضح لنا جميعا فائدة عدد من المجالات الدورية على الرغم من أن الكتب مرجع راجع ومفيد إلا أن المجلات الدورية تعرض لأحدث التقنيات والتغييرات التي تطرأ عليها بالإضافة فإنها تعرض للعديد من الآراء ووجهات النظر ومن هذه المجالات، يمكنك الرجوع كما يلي:

- صحيفة مبرمجي فجوال بيسك ww.devx.com.
- صحيفة نظم مايكروسوفت (www.microsoft.com./msj/.
- مجلة أوراكل www.orgmag.com يمكنك الحصول على اشتراك مجاني.
- تصميم وبرمجة قاعدة البيانات www.dbpd.com وجاري إبدالها بـ المشروعات الذكية.

مواقع الويب

تعد شبكة الإنترنت حيوية وهامة في اتصال الأشخاص بأحدث التقنيات بالإضافة للسماح لك بتحميل ونسخ بعض نماذج رموز البرامج. توجد العديد من المواقع المخصصة الفيجوال بيسك وعدد من المواقع لأوراكل.

- من أهم مواقع الويب هو صفحة كارل وجاري عن فيجوال بيسك Cary and Gary'sVisual Basic Home Page أخرى وتستطيع أنت إضافة أي رابط أخر خاص بك وهذه الصفحة موجودة في http://www.cgvb.com.

- موقع الويب الخاص بمايكروسوفت وهو http://www.microsoft.com.
- موقع الويب الخاص بأوراكل وهو http://www.oracle.com.

المحادثة الإخبارية

تُعد هذه المجموعات هامة في طرح الأسئلة الفنية على كل المهتمين أو العاملين في مجال الكمبيوتر. ولا يوجد أي ضمان لحصولك على أي إجابة للأسئلة التي توجهها أو عند حصولك على الإجابة أن تكون هذه الإجابة أن تكون هذه الإجابة إجابة صحيحة ولكن المؤكد هو حصولك على بعض معلومات كافية لتستطيع البدء في عملك. وفيما يلي بعض الأسماء، لاحظ أنه يجب أن يكون لك صلاحية الوصول لخادم الأخبار من خلال الإنترنت.

- Comp.database.Oracle.misc
- . Comp.database Server
- Comp.lang.basic visual. Database
- Comp.lang.basic visual. Misc
- Microsoft.public. database.rdo

ملاحظة: يوجد في موقع الويب carl and Gary's Web سبل للعديد من المواقع الخاصة بفجوال بيسك في http://www.cgvb/gcgi/news_form.

مساعدة فنية

يوجد في قسم الدعم الفني بأوراكل عدد من الموظفين ذوي الخبرات ومعرفة في أغلب تقنيات مايكروسوفت، ويوجد مثلهم لدى أوراكل. وعادة تتحمل التكاليف مقابل هذا الدعم وهي تكاليف ليست كثيرة خاصة إذا عرفت مقدار الوقت الذي يمكن أخذه في محاولة إصلاح شيء أو معرفة سبب العطل به حتى بعد إصلاحه. لم يكن هذا الكتاب ليظهر بدون مساعدة مسئولي الدعم الفني في هذه الشركات.

- http://support.oracle.com/Metalink تغطي الأسئلة عن العديد من الموضوعات المرتبطة بأوراكل.
- Technical Forums for ODBC and OracleObjects for OLE at http://support.oracle.com

- http://support.microsoft.com/support
- Oracle's new Technology Network for Developers at http://technet.oracle.com

موارد أخرى

على مدار السنوات، وجدنا أن خدمات مايكروسوفت Technet و Microsoft Developers Network (MSDN) مصادر هامة وقيمة للمعلومات وتوجد تكلفة واجبة السداد لهذه الخدمات الموفرة على أقراص مضغوطة، ويمكنك أيضاً الوصول للمعلومات في موقع الويب الخاص بمايكروسوفت.

خاتمة

أخذ هذا الكتاب مجهود كبير وكان مطلوباً لأن المعلومات المدرجة به صعبة الجمع. لقد بذلنا كل ما في وسعنا من جهد لتقديم معلومات كاملة ودقيقة لكل العاملين في التطوير باستخدام فجوال بيسك وأوراكل. ويمكن إرسال أي تعليقات سواء إيجابية أو سلبية وأي اقتراحات من خلال موقع الويب "www.Sybex.com"، وبمجرد الوصول لهذه الصفحة، اختر نموذج Book Content Issues From.

الجزء الأول

التعريف بأوراكل

- أوراكل وفيجوال بيسك
- بنية قاعدة البيانات لأوراكل
- تكوين قاعدة البيانات
- إدارة قاعدة البيانات
- تصميم قاعدة البيانات
- تنفيذ [illegible]

١

أوراكل وفيجوال بيسك

يُعد كل من تطبيق أوراكل Oracle وتطبيق فيجوال بيسك تطبيقاً مختلفاً عن الآخر كما أنهما من مؤسستين مختلفتين وتم تصميمهما للتعامل مع مشكلات مختلفة. وعلى الرغم من ذلك فالمطور الذي يستخدم فيجوال بيسك مطالب بكتابة بعض التطبيقات التي تعمل متصلة مع قاعدة بيانات أوراكل، ولذا نعرض لك في هدا الكتاب مفاهيم وتفاصيل تشغيل وأمثلة لكتابة التعليمات البرمجية لكي تتمكن من فهم البرنامجين بل واستخدامهما في كتابة أكثر التطبيقات قوة وكفاءة.

في الفصل ١ ستبدأ بمقدمة عن أوراكل واستخدامات قاعدة بيانات أوراكل مقابل قاعدة بيانات فيجوال بيسك. ومن الموضوعات التي سنناقشها:

- نبذة عن تاريخ شركة أوراكل
- بنية قاعدة البيانات العلائقية
- خادم أوراكل
- خصائص خادم أوراكل المتقدمة
- فيجوال بيسك في عالم أوراكل
- الانتقال لأوراكل

نبذة عن تاريخ شركة أوراكل

بدأت مؤسسة أوراكل المعروفة الآن والتي تُعد من أكبر شركات صناعة البرمجيات (برامج الكمبيوتر) في العالم، بدأت في عام ١٩٩٧ كشركة صغيرة باسم Relational Software Incorporated (RSI). وأخذ لاري أليسون وآخرين من مؤسسي الشركة فكرة هذه الشركة من العمل الجاري في معامل البحث في شركة IBM لتطوير أول قاعدة بيانات علائقية في العالم وتسمى أوراكل. تم التخطيط لجعل قاعدة البيانات هذه متوافقة مع لغة SQL الجديدة لتحقيق الاتصالات بين قاعدة البيانات هذه وبين التطبيقات الخارجية. واستخدمت لغة C في كتابة برنامج قاعدة البيانات بهدف إمكانية تشغيله في كل الحاسبات والأنظمة. وتحققت هذه التوقعات خلال ٢٠ عام من التطوير كما ساهمت بشكل فعال في نجاح قاعدة بيانات أوراكل.

الإصدار الثاني من أوراكل هو الإصدار الذي تم طرحه بالأسواق لأن الإصدار الأول كان نموذجاً أصلياً فقط، وشهد الإصدار الثالث تغيير اسم الشركة من RSI إلى مؤسسة أوراكل Oracle Corporation. وعلى الرغم من نمو ثقة العميل وإمكانية الاعتماد على المنتج بثبات، إلا أنه لم تصل بنية قاعدة البيانات إلى شكلها الحالي المعروف والموثوق فيه إلا مع الإصدار الخامس، وهو المزود/الخادم "Client/ Server". كما شهد هذا الإصدار تطور في خيار الخادم المتوازي Parallel Server. واستمر الإصدار السادس والسابع في الاتجاه نحو تحسين الأداء كما زادت فيها القابلية للتحجيم.

مع الإصدار الثامن "أوراكل ٨" حدث تحول في تصميم بنية قاعدة البيانات. وكان هذا التحول في اتجاه مفهوم بنية الحاسبات للشبكة، مع قاعدة بيانات أوراكل كجزء أساسي منها. على الرغم من ظهور عدة سمات وخصائص للإصدار الثامن عن الإصدار السابق لـه، الإصدار السابع لأوراكل، إلا أن الإصدار الجديد اتجه بشكل أقرب للكمال وظهر فيه نموذج لقاعدة بيانات عالمية ومثالية، قاعدة بيانات تخزن وتعالج بيانات من أي نوع وليس الجداول العلائقية فقط. ويظهر هذا المفهوم من خلال قدرة أوراكل ٨ على تخزين عدد من الوحدات الكبيرة "Large Objects LOB" ووحدات هيكلية مع الخيار Object وتخصيص لتصنيف الكارتريدج الذي يسمح للمستخدم بمعالجة أنواع البيانات متعددة الأشكال هذه ومنها المعلومات الرسومية (graphical) أو المتحركة (Video). وفي نفس الوقت تم الحفاظ على الاتجاه الذي يهدف لزيادة عدد المستخدمين والبيانات وتحسين الأداء.

بنية قاعدة البيانات العلائقية

عندما يذكر اسم شركة أوراكل، يرتبط ذلك بنظام إدارة قاعدة بيانات علائقية قوي وبارع. في الفصل الثاني، سنعرض لهيكل "RDBMS" وهو اختصار للمصطلح Relational Database Management System وهو نظام إدارة قاعدة البيانات العلائقية كما سنعرض للجداول والأعمدة والعلاقات بين الجداول التي تتعلق بـ RDBMS. ومع ذلك يوجد موضوع وثيق الصلة بالبنية الأساسية لـ RDBMS والذي يفصل بين النماذج البسيطة والنماذج المتقدمة. في هذه الجزئية ستتعرف على ثلاثة أنواع من أنواع البنية الأساسية وهي: الأنظمة التي تعتمد على الملفات File-based وأنظمة المزود/الخادم Client/Server والبنية متعددة الطبقات Multitier. كما سنعرض لتنفيذ البنية متعددة الطبقات في أوراكل وهي بنية الحسابات للشبكة (NCA). يجب معرفة الفرق بين أنواع البنيات المختلفة بحيث تتعرف على سبب الحاجة للانتقال من نوع لآخر في حالة زيادة سعة أو تعقيد التطبيقات أو قواعد البيانات المستخدمة.

أنظمة تعتمد على الملفات "File-based"

أبسط مثال ونموذج لقاعدة البيانات العلائقية ممثل في منتجات مثل مايكروسوفت أكسس. ويتكون أكسس من ملف واحد له الامتداد "mdb." ويتضمن هذا الملف كل جداول واستعلامات ونماذج قاعدة البيانات التي تنطبق على قاعدة البيانات وأي معلومات إقفال أخرى. يمكن وضع الملف في كمبيوتر المستخدم أو في أي مكان على الشبكة.

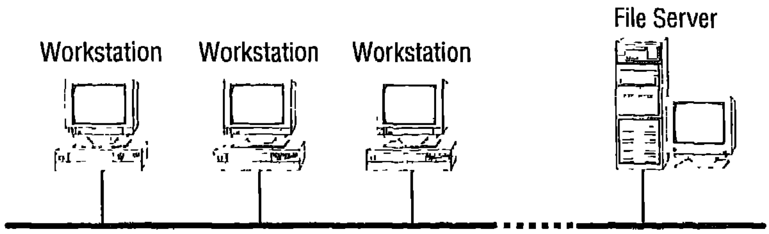
على الرغم من استخدام برنامج أكسس في عرض النماذج والاستعلامات الخاصة بملف قاعدة البيانات، إلا أنه يمكنك أيضاً استخدام أي برنامج آخر مثل فيجوال بيسك، لعمل ذلك. يجب أن

يتعرف كل تطبيق مستخدم على كيفية قراءة الجداول أو الكتابة فيها في قاعدة البيانات، وأيضاً كيفية استخدام نظم إدارة الإقفال بحيث يستطيع أكثر من مستخدم استخدام قاعدة البيانات فى وقت واحد، كما يجب أن يزيد على تفهم الوسيط للتنسيق التام بينه وبين قواعد صلاحيات الوصول. في حالة أكسس يتم تحميل الوسيط بواسطة المحرك JET. وفي حالة استخدامك لأي طريقة أخرى للوصول للبيانات، قد يؤدي ذلك إلى إتلاف قاعدة البيانات.

يوضح الشكل ١-١ قاعدة بيانات أكسس مرتبطة بعدد من المستخدمين على الشبكة. وفى هذا الشكل، يتم وضع قاعدة البيانات على خادم الملفات ولا توجد أي قاعدة بيانات فعالة على الخادم، ولذا يتعرف التطبيق على كيفية استخدامها.

الشكل ١-١

بنية قاعدة البيانات تعتمد على الملفات.



ونظهر مشكلات هذا النظام في القابلية للتحجيم. فالنظام الذي يعتمد على الملفات يكون أكثر ملاءمة لمجموعات العمل الصغيرة. وعلى الرغم من تعامله بنجاح مع بيانات سعتها حوالي ١٠٠ ميجابايت، إلا أن أداءه يقل عند تعامله مع أكثر من ١٢ مستخدم وأيضا عندما يزيد عدد السجلات في الجداول. على سبيل المثال، إذا أردت البحث عن أعلى قيمة لعمود في مليون سجل، فسيقوم النظام المعتمد على الملفات باستعادة كل السجلات الموجودة على الشبكة ثم معالجتها داخليا (محليا) وأخيرا إيجاد القيم المطلوبة.

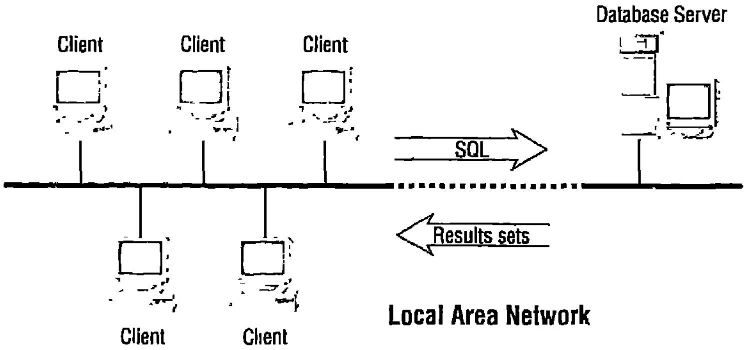
بنية المزود/الخادم "Client/Server"

مع زيادة المتطلبات ومع زيادة البيانات والمستخدمين، نجد أن النظام المعتمد على الملفات يصل إلى مرحلة من البطيء في الاستخدام. ويرجع ذلك لحجم السجلات وعدد المستخدمين وسرعة الشبكة، ولذا يجب البحث عن طريقة لمعالجة البيانات على كمبيوتر سريع للغاية لتحويل أقل قدر من المعلومات عبر الشبكة اللازمة لأداء المهمة المطلوبة. وفي حالة مثل حالة البحث في مليون سجل، يجب استعادة قيمة واحدة فقط هي القيمة المطلوبة. ومن هنا تظهر الحاجة لبنية المزود/الخادم، وتظهر أيضا أهمية منتجات مثل أوراكل.

يعرض الشكل ١-٢ بنية المزود/الخادم ومكوناتها من المزود والخادم. والمـزود فـي هـذه الحالة هو المستخدم الفردي الذي يقوم بتشغيل تطبيقات مثل فيجوال بيسك، أما الخادم فهو جـهاز قوي في المؤسسة يقوم بتشغيل خادم فعال لقاعدة بيانات.

الشكل ١-٢

بنيـــة المزود/الخادم.



في هذا الكتاب سنعرض لتطبيقات المزود المكتوبة بفيجوال بيسك في نظـام مايكروسوفت ويندوز ٩٥ أو ويندوز NT. ولكن يمكن تكوين هذه التطبيقات بعدة لغات وفي أجـهزة كمبيوتـر وأنظمة تشغيل مختلفة. وفيما يتعلق بالبرامج المستخدمة على الخادم، سنعرض بالتفصيل لنظـام أوراكل الذي يمكن وضعه على أي جهاز وأي نظام تشغيل.

ما هي مميزات استخدام بنية المزود/الخادم؟ تكمن أغلب المميزات فـي القـدرات المتعـددة لمعالجة البيانات. في حالة وجود كمبيوتر موصل بالشبكة، يفضـل الاسـتغناء عـن النـهايات الطرفية. يكون للحاسب الشخصي الحديث قدرة هائلة على معالجة البيانات وعرضها في العديـد من الأشكال الرسومية التي أصبحنا ننتجها بواسطة تطبيقات سطح المكتب. وبذلك لا يخصـص الخادم أجزاء من المشغل لتنسيق البيانات وإعداد التقارير، وبدلا من ذلك فهو يقوم ببساطة بنقـل البيانات للمزود. وبذلك تنقل الشبكة بيانات أقل ولكن ليس هذا هو الحال دائماً.

ما هو المزود

من وجهة نظر أوراكل. يتم التعامل مع الجزء الخاص بالخـادم فـي البنيـة. ولكـن ما هو المزود؟ في بنيات قاعدة البيانات الأولية، كـانت تطبيقـات قـاعدة البيانـات على كمبيوتر كبير يعـرف باسـم mainframe. وكـان لكـل مسـتخدم صلاحيـة الوصول للتطبيقات من خلال النهايات الطرفية الموجـودة علـى الشـبكة، وكـانت أحيانا تسمى (dumb) لأنها أدوات عرض فقـط ولا تمتلـك أي قـدرات لمعالجـة البيانـات.

مع تقدم وانتشار أجهزة الكمبيوتر الشخصية، ظهرت الحاجة لأجهزة سطح المكتب والتي تستطيع معالجة البيانات محليا والتى بدورها تخفف عبء المعالجة على الخادم وكان البديل للنهايات الطرفية هو المزود الموجود حاليا.

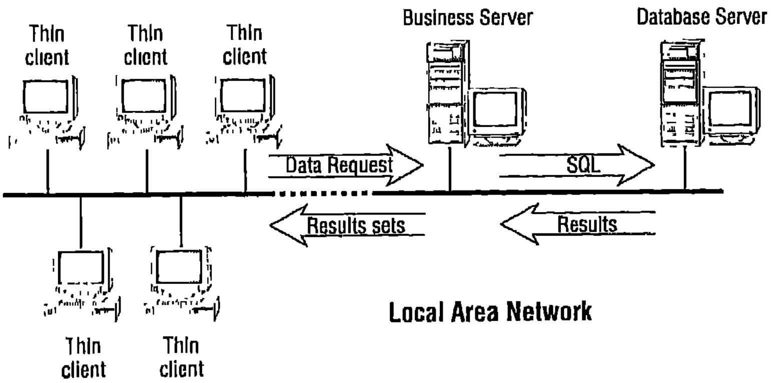
أما برامج المزود فتم إبدالها بتطبيقات مكتوبة بفيجوال بيسك وغيرها. مع المرونة والشكل الرسومي المبهر لتطبيقات فيجوال بيسك، يجب أن نتحمل مسئولية إدارة بداية الوصلات، وطلب واستعادة البيانات من قاعدة البيانات على الشبكة وتفسير وعرض هذه البيانات بطريقة محكمة ويعتمد عليها.

البنية متعددة الطبقات Multitier

يوجد في البنية القياسية للمزود/الخادم بعض المشكلات فى القابلية للتحجيم (scalability) وتم إيجاد العديد من الحلول لها. والمصطلح multitier يعني وجود العديد من طبقات البرامج لكل منها مسئولية محددة لمهمة معينة. وكما نرى فى الشكل ١-٣، تنقسم الطبقات في أغلب الأحوال إلى ثلاثة أجزاء: المزود أو طبقة الوسيط الرسومي GUI والطبقة الوسطى أو طبقة الأعمال وأخيراً طبقة البيانات. وتعرف أيضاً على البنية بالبنية ثلاثية الطبقات (three-tier) ولكن هذه التسمية تخفي إمكانية انتشار هذا البرنامج في العديد من أجهزة الكمبيوتر ويكون لكل جهاز منها دور صغير في المهام الأساسية.

الشكل ١-٣

البنية متعددة الطبقات.



وفي هذه الطريقة تلعب تطبيقات فيجوال بيسك دوراً ما في المزود، مثل دورها في بنية المزود/الخادم المعروفة، على الرغم من أنها ستتضمن معالجات أقل للبيانات وتركيزاً أكبر على خاصية العرض وتلعب فيجوال بيسك دوراً متزايداً في الطبقة الوسطى التي يتم فيها التقاط وتوجيه قواعد عمليات المعالجة وقواعد الأعمال. وأصبح هذا الدور متاحاً لأن باستطاعة فيجوال

بيسك تكوين مكونات برامج ActiveX التي يمكن تشغيلها على الخادم في بيئة لمعالجة الحركات مثل Microsoft Transaction Server (MTS).

تم تصميم البنية متعددة الطبقات لإيجاد حلول لمشكلات المزود/الخادم. ومن الخصائص الموجودة فيها ما يلي:

- سهولة نشر التطبيقات مع الاحتفاظ ببنية متعددة الطبقات ويُعد هذا هاماً للغاية عندما تريد تحجيم التطبيق للاستخدام على الويب.
- قواعد الأعمال في مزود "thick" تعني وجوب تبديل تطبيق المزود في كل مرة يتم تغيير قواعد الأعمال فيها.
- عند تغليف كل قواعد البيانات ووضعها في الطبقة الوسطى، سيستخدم أي تطبيق يحتاج للوصول إلى هذه القواعد مثل هذه المكونات المشتركة بدلاً من وضع نفس القواعد في كل تطبيق موجود بالمزود.
- تضمن لك طبقة الأعمال سرية البيانات بطريقة معروفة وتضمن ثبات التطبيقات لأكبر قدر ممكن.
- توجد حلول لمشكلات قابلية التحجيم. على الرغم من وجود بعض الإضافات للتطبيقات الصغيرة، فإن المزايا تكمن في زيادة الحجم لأنه يمكنك بسهولة تضمين أجهزة أكثر للطبقة الوسطى.

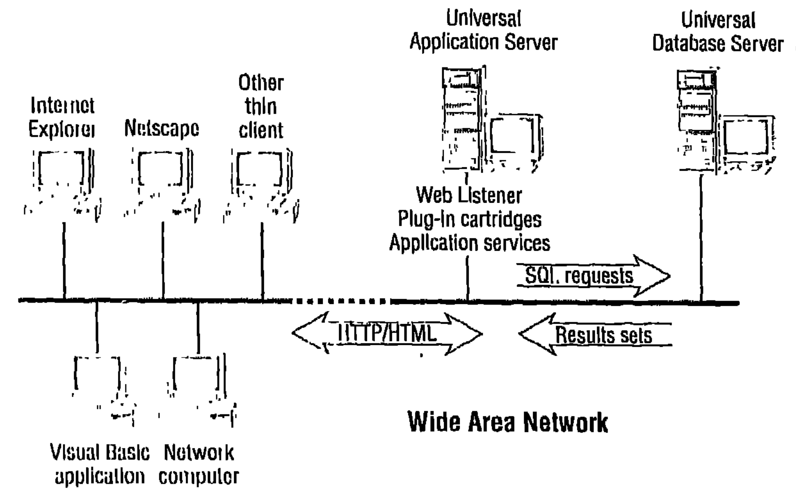
ملاحظـة. تعمل الطُرق المشروحة في هذا الكتاب بنفس الكفاءة في نظام المزود/الخادم أو عندما يتم وضعها في مكونات تعمل في الطبقة الوسطى.

بنية الحاسبات للشبكة NCA

يسمى تطبيق أوراكل للتصميم المتعدد الطبقات ببنية الحاسبات للشبكة Network Computing Architecture وتتبع نفس الخصائص الأساسية الموجودة في الشكل ١-٣. وعلى الرغم من ذلك نجد أن NCA موجود بشكل أكبر في الطبقات الوسطى والنهائية مع تقليص دور جانب المزود لدور ثانوي في معالجة البيانات. وكما هو الحال في البنية متعددة الطبقات تتضمن NCA ثلاثة طبقات أو أكثر يمكنك رؤيتها في الشكل ١-٤.

الشكل ١-٤

بنية الحاسبات للشبكة NCA.



تقع كل طبقة من الطبقات الثلاث في فئة من فئات التصنيف المشروحة. فيما يلي ستلاحظ أن منتجات مثل فيجوال بيسك (التي تكمن قوتها الحقيقية في الوسيط المرئي) ليس لها دور في الوقت الحالي. يجب تغيير ذلك في المستقبل لأن أوراكل يتوسع في أنواع المكونات (أو الوحدات) التي يمكن وضعها في طبقة خادم التطبيق الوسطى.

المزود

المزود هو مزود "عالمي" بسيط وقد يكون واحد مما يلي:

- مستعرض ويب تقليدي
- مزود يعتمد على جافا "Java"
- كمبيوتر على الشبكة

والهدف من استخدام مزود بسيط هو جعل التطبيقات متاحة لأي مستخدم لديه نظام تشغيل أو برنامج وصول من اختياره. ويستخدم المزود كواجهة رسومية نهائية لها قدرة معالجة محدودة. لاحظ أن التصميم المتعدد الطبقات العام لا يحدد مزوداً بسيطاً. يمكنك وضع برامج الطبقة الوسطى أو النهائية على المزود ويرجع ذلك للتصميم. وتلعب بنية مايكروسوفت دوراً أكبر من تقديم الخدمات التي يمكن الجمع بينها على جهاز واحد عند الحاجة. أما NCA لأوراكل فهي أكثر حرصاً والتزاماً بإبعاد معالجة الأعمال عن المزود. وعلى الرغم من ذلك، لا يوجد أي سبب فني لوقف استخدام تطبيق فيجوال بيسك كمزود.

خادم التطبيقات

مصطلح "خادم التطبيقات" مصطلح غير معروف للجميع، فهو تنفيذ لبرنامج متخصص مع بعض أجزاء الكمبيوتر Hardware ذات الصلة التي توفر قاعدة فعالة وقابلة للتحجيم للطبقة الوسطى. وأحد أهداف أي برنامج في الطبقة الوسطى هو ضمان إمكانية استخدام أي تعليمات برمجية أو أي وحدة أو أي مكون – بواسطة أي طلب يدخل إلى الخادم. في السابق ومع سكريبتات CGI لخادم الويب، كان الخادم يقوم ببدء عملية جديدة لكل طلب. إلا أن سكريبتات CGI لم تثبت قابليتها للتحجيم. أما باستخدام خادم التطبيقات، يمكن استخدام أي نوع من البرامج المتوافقة في نفس مساحة المعالجة، ويمكن تجميعها لكل طلبات المستخدم. لخادم تطبيقات أوراكل، تعتمد الوحدات على CORBCA وهي Common Object Request Broker Architecture. والمناظر لها في مايكروسوفت هو استخدام، Microsoft Transaction Server الذي يعتمد على Component Object Model والتي نعرفها باسم COM. في الوقت الحالي، تُعد البنية CORBA والبنية COM غير متبادلتين على الرغم من وجود جهود للربط بينهما. ولذا يمكنك كتابة مكونات ActiveX لـ MTS وليس لخادم التطبيقات لأوراكل.

يُعد خادم التطبيقات تصميماً مرناً بسبب ما يسمى في أوراكل بوحدات الكارتريدج Cartridges وهي المنتجات التي يمكن تشغيلها على برامج الخادم بنفس طريقة البرامج المضافة الموجودة على مستعرض الويب الذي تفضل استخدامه. باستخدام كارتريدج، فإنك تقوم أساسا بتوسيع برامج التطبيقات الأساسية بواسطة التعليمات البرمجية الخاصة بك. ومن مميزات الكارتريدج أنه عند حدوث أي خطأ أو عطل في التعليمات البرمجية، فلن يتسبب ذلك في إقفال خادم التطبيقات وتوقفه عن العمل، ولذا يمكنك الاعتماد عليه. والآن توجد وحدات كارتريدج للبرامج المضافة لكل من Java وPL/SQL وPERL وC.

خادم البيانات العام

عادة يشار لطبقة خادم البيانات باسم خادم البيانات العام Universal Data Server لأن تصميم قاعدة البيانات هذه يهدف لمعالجة أي نوع من أنواع البيانات. ولقد تم تصميم أوراكل٨ لتحقيق هذا الهدف من خلال شيئين هما الخصائص والقدرة على التوسع. تعد مجموعة الخصائص المتوفرة في أوراكل٨ من أغنى ما تم إصداره حتى الآن، حيث يجب ملاحظة قدرتها على معالجة البيانات حتى ٤ جيجابايت. وتعد هذه القدرة مفيدة خاصة للرسوميات وصور الفيديو (المرئية). على رغم من القدرات الداخلية المتضمنة (built-in) في خادم البيانات العام، إلا أن قدرته على التوسع قد تشكل خطرا على استمرار نجاحه. يمكن تحقيق القدرة على التوسع من خلال نفس مفهوم الكارتريدج الذي شرحناه في الجزئية الخاصة بطبقة خادم التطبيقات.

أنواع كارتريدج التي يمكن استخدامها حاليا هي:

- كارتريدج PL/SQL لتكوين الإجراءات المخزنة.
- كارتريدج ConText لقدرات المراجعة والبحث المتطور في النصوص.
- كارتريدج Image/VIR لإدارة الرسوم المحلية لدعم التطبيقات متعددة الوسائط.
- كارتريدج مساحي Spatial لإدارة ومعالجة المعلومات الجغرافية.
- Time Series لصيانة والاستعلام عن البيانات المعتمدة على الوقت.

The COM Cartridge

تزامن الإعداد لـهذا الكتـاب مـع الإعـلان عـن ظـهور نسـخة تجريبيـة مـن كارتريدج COM متوفر كـ DLL لوينـدوز NT ويسـمح لـك باسـتخدام التشـغيل الآلي البعيـد Remote Automation للتحكـم فـي وحـدات التحكـم COM مـن داخل أوراكل٨. على سبيل المثال، يمكنك التحكـم فـي طـرق وخصـائص إكسـل وورد من الإجـراءات المخزنـة. ويتوقـع أن تسـتطيع تكويـن وحـدات COM الخاصة بك (نتمنى أن يحدث ذلك مـع فيجـوال بيسـك) وذلـك لزيـادة إمكانيـات قاعدة البيانـات.

خادم أوراكل

عند تفحص طبقة خادم البيانات الخاصة بـ NCA عن قرب، نجـد أن لأوراكـل سلسـلة مـن المنتجات القوية والقابلة للتحجيم والتوسيع. قامت أوراكل بتصميم قاعدة بيانات عاليـة الجـودة والتي ازدادت في النمو خلال العشرين سنة الماضية، وبذلك حققت حجم الأداء وعوامـل الثقـة والإتاحة المتوقعة من أداة تعمل على مستوى المؤسسة.

على الرغم من تغيير أسماء المنتجات أكثر من مرة خلال السنوات القليلـة الماضيـة إلا أن الشكل الحالي هو:

خادم أوراكل٨: قاعدة بيانات اقتصادية تعمل على مستوى المجموعات وتتضمن الخصـائص الأساسية ولكنها تفتقر القابلية للتوسع الموجودة في إصدارة المؤسسات.

إصدارة المؤسسات من أوراكل٨: قاعدة بيانات كاملة تعمل على مستوى المؤسسة وتسمح لك باستخدام خصائص متقدمة مثل إمكانية استخدام أكثر من وحدة معالجة مركزيـة CPU مـع التطبيقات المتوازية، واستخدام وحدة الكارتريدج المتخصصة للبيانات. يتضمن هذا الإصـدار النسخ المماثل المتطور (Advanced Replication)، وفي مقابل بعض التكلفـة الإضافيـة، يمكنك الارتقاء والعمل بواسطة خادم أوراكل المتوازي.

Oracle Lite: قاعدة بيانات صغيرة وشخصية مناسبة لعمليات الكمبيوتر التي تتم عن بعد.

خصائص قياسية

كما نتوقع، توجد في أوراكل قاعدة بيانات على مستوى المؤسسة تتضمن عروض الخصائص القياسية. فيما يلي قائمة لهذه الخصائص وشرح مفصل لها في الأجزاء القادمة من الكتاب (في الجزء الأول والثاني).

- البنية العلائقية
- أدوات الإدارة
- إدارة السرية
- إمكانات عمل النسخ الاحتياطية والاستعادة
- واجهةSQL للاستعلامات وعمليات التحديث
- إدارة الاتصالات للتطبيقات الخارجية
- الزنادات والإجراءات المخزنة

على الرغم من تمييز أوراكل بالخصائص المتقدمة التي سنعرض لها فيما يلي، إلا أن النسخة القياسية تتميز بشكل أكبر في نقطتين وهما القابلية للتحجيم والأنظمة المتعددة.

القابلية للتحجيم

يستخدم هذا المصطلح (Scalability) خاصة عند الحاجة لتكوين أنظمة تستطيع التعامل مع مزود يعتمد على الويب يتطلب إمكانية كبيرة وأي عدد من المستخدمين. وفي الواقع، تعتمد فكرة القابلية للتحجيم على اتجاه كل فرد، أي تختلف باختلاف الشخص واتجاهاته. ومن بعض النقاط الأكثر أهمية:

- يمكنك البدء من أرخص أجهزة الكمبيوتر والتدرج إلى الأحدث والأقوى من هذه الأجهزة مع ظهور الحاجة لذلك.
- تستطيع اقتناء حاسبات وأنظمة متنوعة داخل الشركة الواحدة، واختيارها وفق متطلبات مجموعات العمل. فقد تحتاج إدارة البحث والتطوير لقاعدة بيانات صغيرة بينما تحتاج إدارة الحسابات والماليات لقاعدة بيانات على كمبيوتر كبير mainframe.
- لا يجب التقيد بمنتجات شركة واحدة سواء لأجزاء الكمبيوتر أو للبرامج وعلى أسوأ الفروض، يمكنك تحويل قاعدة البيانات الخاصة بك من النظام الحالي إلى أي نظام آخر بمنتهى السهولة.

- لا يجب التقيد بشراء أجهزة جديدة إذا كان لديك خادم مناسب لك، بحيث تستطيع الاحتفاظ بمثل هذه المصروفات لبنود أخرى مثل شراء معدات جديدة أو توجيهها لزيادة خبرات العاملين بشركتك.

يجب وضع موضوعات دعم الأنظمة المتعددة في الاعتبار. لن تستطيع مجموعات تطوير قواعد البيانات العمل في عدة أنظمة في وقت واحد. يوجد بأوراكل نظم تستخدم كمرجع (reference) يتم تطوير قاعدة البيانات فيها قبل نقلها إلى نظم أخرى. ومن النظم المستخدمة كمراجع نظامية هما Unix وويندوز NT. إذا كنت تستخدم نظام تشغيل آخر، قد تضطر للانتظار عدة أشهر قبل أن تظهر أحدث نسخة على نظام التشغيل الخاص بك.

ملاحظة على الرغم من إمكانية تشغيل خادم قاعدة البيانات على عدة أنظمة، فليست كل أدوات الدعم بها نفس الخاصية. ويظهر في NT أغلب أدوات الإدارة مثل Enterprise Manager.

خصائص خادم أوراكل المتقدمة

يهدف هذا الكتاب في المقام الأول إلى توفير مقدمة عن أوراكل ولا يهدف إلى تنفيذ خصائص متقدمة في أوراكل. وعلى الرغم من ذلك، ففي الأجزاء التالية سنعرض لعدد قليل من الخصائص المتقدمة التي قد تصادفها عند تثبيت إصدارة المؤسسات. يجب الحذر عند استخدام الخصائص المتقدمة، لأنه على الرغم من دعمها لديك إلا إنها ستعمل معها كمطور لفيجوال بيسك.

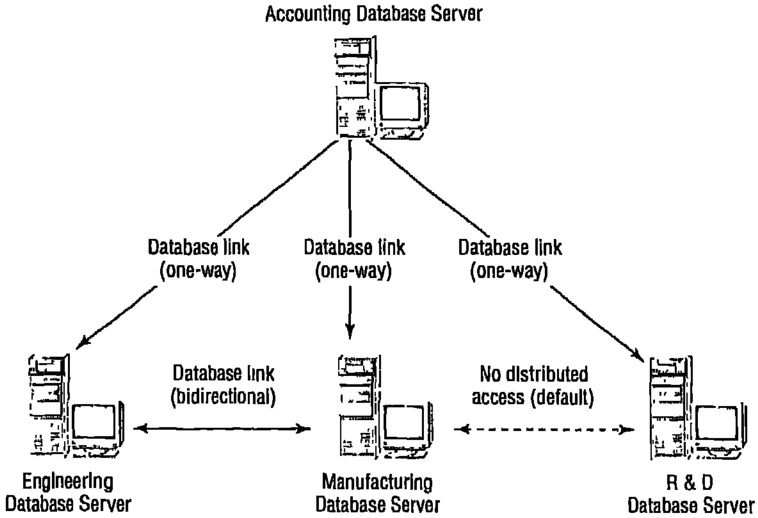
تكمن الاستفادة الكبرى من هذه الخصائص في تحسين الأداء والقابلية للتحجيم وإتاحة قاعدة البيانات، وهي خصائص قد لا تجدها في قاعدة بيانات غير أوراكل. على الرغم من ذلك، تتطلب هذه الخيارات كما مناسبا من التخطيط والتدريب في حالة خادم أوراكل المتوازي Oracle Parallel Server. يوجد أيضاً زيادة ملحوظة في التكلفة بالمقارنة بإصدارة المؤسسات القياسية.

قواعد البيانات الموزعة

تتضمن أوراكل خيار Distributed في كل خادم على مستوى إصدارة المؤسسات. يسمح لك هذا الخيار بإعداد روابط بين أي قاعدتي بيانات على نفس الشبكة. على سبيل المثال، قد تحتاج إدارة الحسابات للوصول إلى بيانات في أجزاء أخرى من الشركة، مثل الإدارات التي يكون لكل منها قاعدة بيانات خاصة بها. لتحقيق ذلك يمكنك بسهولة عمل رابط من ناحية واحدة من قاعدة بيانات الحسابات إلى قاعدة بيانات التصنيع وإدارة البحث والتطوير والإدارة الهندسية كما هو موضح في الشكل ١–٥.

الشكل ١-٥

روابط قاعدة البيانات في قاعدة البيانات الموزعة.



في المثال، يستطيع أي مستخدم فى إدارة الحسابات تقديم طلب من أي قاعدة بيانات تم إعداد الرابط معها. والطريقة سهلة للغاية في سهولة إضافة موقع جدول عند تقديم الاستعلام. كما بمكنك تكوين رابط بيانات واضح باستخدام المترادفات. توجد بعض الإجراءات الأمنية المتنوعة التي نضمن لك عدم استخدام أي مستخدم لهذه الروابط إلا إذا كان لديه كلمة السر الصحيحة التي تسمح له بذلك.

من مميزات قاعدة البيانات الموزعة، إمكانية الفصل سواء الفصل الفعلي أو الفصل الوظيفي للبيانات. كما أنه عند حدوث أي عطل في أي قاعدة بيانات، لن يتسبب ذلك في تعطيل أو إيقاف قواعد البيانات الأخرى (ولكن في هذه الحالة لن يتم تشغيل الاستعلامات الموزعة distributed queries). كما يمكنك الوصول إلى قواعد بيانات أوراكل الموجودة في أنظمة تشغيل مختلفة إذا كانت موجودة على نفس الشبكة، ولا يجب أن تكون إصدارات أوراكل متماثلة.

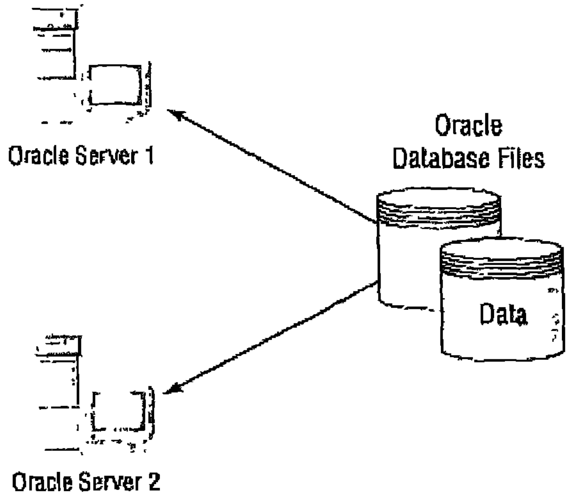
الخادم المتوازي

تُعد RDBMS التقليدية عملية تطبيق واحدة أو أكثر ترتبط بها ذاكرة مقابل مجموعة من ملفات قاعدة البيانات. وتقدمت أوراكل نحو المستوى التالي بخيار أطلقت عليه اسم Parallel Server أي الخادم المتوازي. يقوم هذا الخيار بتمكين وصول أكثر من خادم لنفس ملفات قاعدة البيانات (انظر الشكل ١-٦). يجب على التقنيات المركبة (مثل مدير الإقفال الموزع Distributed Lock Manager) تمكين هذا الخيار بحيث يستطيع المستخدمون في كل خادم إجراء التغييرات على نفس البيانات كما هو الحال مع وجود خادم واحد. يجب الجمع بين كل خادم والآخر، بحيث يكون Unix و OpenVMS هما نظامي التشغيل التقليديين، ولكن خيار Parallel Server بدأ في الدخول

إلى نظام NT أيضاً. في أغلب الأحوال يتم وضع خادمين متوازيين ولكن يمكن وضع عشرات بل مئات من الخدام.

الشكل ١-٦

خاصية الخادم المتوازي في أوراكل.



ومن مميزات الخادم المتوازي:

- **الأداء:** يوفر لك قدرة تشغيل وإمكانية زيادة المخرجات في مقابل تكلفة بسيطة.
- **الثقة في الاعتماد عليه:** في حالة تعطل أي خادم لأي سبب من الأسباب، تنتقل تطبيقات المستخدم لأي خادم آخر.

الخادم المتوازي لأوراكل ليس مجانيا، أي يتم تحصيل رسوم إضافية عليه، كما يحتاج لتخطيط وتنفيذ دقيق. وستحتاج أيضاً لتدريب إضافي لأنه إذا لم يتم تعريف مكونات الخادم المتوازي بطريقة صحيحة، فلن يكون مستوى الأداء كما هو متوقع. على الرغم من ذلك، إذا كنت تحتاج لنظام تثق وتعتمد عليه وإمكانية زيادة إخراجات فيجب أخذ خيار الخادم المتوازي Parallel Server في الاعتبار.

ملاحظة

الخادم المتوازي لا يحميك ضد تعطل وفشل القرص الصلب hard-disk failure. لذا يجب استخدام RAID disk array ويفضل أيضا النسخ المماثل لقاعدة البيانات أو لقاعدة البيانات الفورية المتاحة لضمان إمكانية الانتقال من قاعدة البيانات لقاعدة أخرى في حالة حدوث أي عطل أو فشل للقاعدة الأولى.

النسخ المماثل

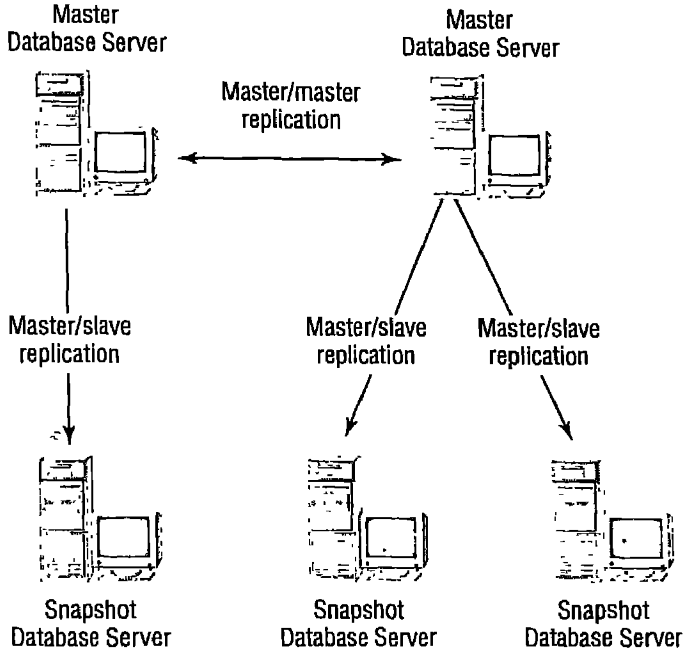
في بعض الأوقات، قد يصبح توافر البيانات وإتاحتها عنصر هام لأعمالك ولا تستطيع الاستغناء عنه إلا لفترة قصيرة، وقد تصل هذه الفترة لدقائق أو حتى ثواني. لذا في حالة خادم قاعدة البيانات الفردي ستتعرض لأوقات لن تكون فيها قاعدة البيانات متاحة، يمكنك مواجهة ذلك والحماية ضد تعطل القرص الصلب باستخدام RAID وهو اختصار لـ Redundant Array of Inexpensive Disks. لكن يحدث خطأ في نواحي أخرى، فقد يتم بدون قصد إقفال أحد المصادر فيؤدي ذلك إلى توقف وعطل في أجهزة الكمبيوتر أو الشبكة. كما أن استدعاء الشخص المناسب والبحث عن سبب العطل سيأخذ الكثير من الوقت.

في المواقف الحرجة والتي تحتاج فيها لحل سريع، نجد أن الحل الحقيقي والمناسب هو وجود قاعدة بيانات أخرى في خادم ثان يمكن جمعها مع الخادم الأول. يكون الخادم الثاني مفيدا للغاية عند تعطل الخادم الأساسي، ولكن مع ذلك سيحدث فشل في الموقع "site failure" وفيه سيكون الموقع بالكامل غير متاح. ولذا يعُد وجود البديل الثاني (أو الثالث) أحد نقاط القوة في أوراكل. وتوجد طريقتين لتوفير قواعد البيانات البديلة، تسمى الطريقة الأولى للنسخة المماثل Replication والثانية قاعدة بيانات فورية متاحة Hot-standby.

في مجال قواعد البيانات قد يعني المصطلح نسخ مماثل العديد من الأشياء، ولكنه في أوراكل يعني أخذ البيانات من قاعدة بيانات وتحويلها إلى أخرى. يمكن تنفيذ ذلك باستخدام لقطات بسيطة في علاقة master/slave والتي يتم نقل البيانات فيها من جهة واحدة فقط والبديل هو علاقة master/master التي يمكن فيها إدخال البيانات في أي قاعدة بيانات ثم يمكن تحويلها بعد ذلك إلى قواعد البيانات الأخرى. يعرض الشكل ١-٧ المواصفات المتاحة.

الشكل ١-٧

المواصفات المتاحة للنسخ المماثل.



فيما يلي نعرض لأنواع النسخ المماثلة المأخوذة في الاعتبار في نسخة أوراكل المماثلة:

Snapshot replication: تؤخذ نسخة دورية من قاعدة البيانات الرئيسية وتستخدم في الإضافة إلى قاعدة البيانات الفرعية أو في تحديثها بالكامل.

Synchronous replication: يتم إدخال البيانات في قاعدتي البيانات في نفس الوقت. في حالة عدم قدرتها على الاتصال ببعضهما البعض، فلن يتم إدخال البيانات. ولكن تظل قاعدتي البيانات دائما متزامنة.

Asynchronous replication: يتم إدخال التغييرات في قاعدة البيانات الرئيسية في طابور متسلسل queue ويتم تحويلها إلى قواعد البيانات الأخرى على أساس دوري. في حالة تعطل أو عدم إقفال الخادم أو الاتصال به، يقوم النظام بتأجيل تحويل البيانات بسبب ذلك قد يحدث خلل في تزامن قواعد البيانات.

تتوافر هذه الخصائص لدى العديد من موردي قواعد البيانات إلا أن أوراكل تتمتع بنقطة قوة تتمثل في معالجة وإدارة التضارب في حالة Replication Asynchronous، قد يحدث التضارب عند تحويل البيانات المحدثة إلى قواعد بيانات أخرى. على سبيل المثال، ماذا يحدث في حالة تحديث صف تم حذفه من قاعدة بيانات أخرى؟ وهنا يسمح لك أوراكل بتحديد مواقع التضارب، بل ويوفر لك عدة طرق لمعالجة هذا الموقف وتستطيع تكوين قواعد خاصة بك لمعالجة مثل هذا التضارب بنفسك.

على الرغم من أنها ليست نسخ مماثلة، فيوجد حل بديل لمشكلة الإتاحة وهو تحويل عمليات التغيير التي حدثت إلى أرشيف على جهاز آخر خاص بالبيانات. ويُسمى جهاز قاعدة البيانات الثانية هذا بجهاز فوري متاح ويعمل في طور يسمى Constant Recovery أي طور الاستعادة الثابت. ويعد هذا الجهاز قاعدة بيانات بسيطة سواء في تكوين أو معالجة الإعداد، ولكن لن تستطيع استخدام الجهاز الثاني إلا إذا حولته ليكون الجهاز الرئيسي "وفي هذه الحالة لن تستطيع استخدام الجهاز الأول".

فيجوال بيسك في عالم أوراكل

تُعد فيجوال بيسك من أكثر لغات البرمجة انتشاراً في العالم، ولكن هل يعني ذلك إمكانية استخدامها كطرف أمامي front end لأوراكل؟ يوجد لدى أوراكل مجموعة عالية الجودة من الأدوات في سلسلة Oracle Developer 2000 بعيداً عن الأدوات الأخرى التي يمكنها عمل ما تقوم به فيجوال بيسك مثل ++C وJava وDelphi. ولا يوجد أي حل بسيط لهذا السؤال، وقد يعتمد ذلك على الاتجاهات والتفضيلات الفردية. على الرغم من ذلك، توجد بعض الأسباب الهامة لاستخدام فيجوال بيسك بدلاً من غيرها من لغات البرمجة ومنها:

القدرة: فيجوال بسك لديه القدرة لإنتاج برامج معقدة ومتطورة مثل غيرها من التقنيات المتوافرة للوصول للبيانات. ولا يوجد الكثير الذي تستطيع إنجازه في Developer ولا يمكن تنفيذه بنفس الكفاءة باستخدام ADO أو OO4O.

المرونة: لا يجب استخدام فيجوال بيسك للوصول لقاعدة البيانات. يمكنك استخدامها في كتابة معالجات النصوص أو برامج التنبيه لوجود بريد إلكتروني e-mail listener أو غيرها من المهام.

الألفة: لأنها أكثر لغات البرمجة انتشارا وشعبية في العالم، فذلك يعني وجود عدد كبير من الأشخاص المدربين الذين يمكنك ضمهم لمجموعات التطوير والدعم في شركتك. فعندما يقل هذا العدد ستجد أنه أيضا أكبر من العدد المتخصص في أي لغة أخرى.

الانتشار: يعد انتشار أي منتج أمرا هاما لأنه يواكب هذا الانتشار وجود موضوعات في المجالات والكتب المتخصصة بالإضافة إلى منتجات أخرى تعتمد عليها وغيرها.

في الماضي

في السنوات القليلة الماضية، لم يكن دعم فيجوال بيسك لأوراكل جيدا بالدرجة الكافية وحتى الإصدار Visual Basic 3، لم يكن هناك دعم لقواعد البيانات البعيدة Remote DB. (في

الجدول ١-١ يوجد ملخص تقريبي للدعم بطرق فيجوال بيسك). وحدات Oracle Objects for OLE (OO4O) هي أحد منتجات شركة أوراكل والتي قامت دائما بدعم عدد كبير من الخصائص خاصة بسياق أوراكل سنعرض لذلك بالتفصيل في الفصل ١٨، "وحدات أوراكل التي تخص OLE". وعلى الرغم من ذلك نجد أن دعم منتجات ODBC أقل.

الجدول ١-١: عرض لدعم فيجوال بيسك لأوراكل للإصدارات المختلفة.

OO4O	ADO	RDO	JET	Versions
OO4O 1.0	N/A	N/A	JET 1.1	VB3
OO4O 2.0	N/A	RDO 1.0	JET 2.5	VB4
OO4O 2.1	ADO 1.5	RDO 2.0	JET 3.5	VB5
OO4O 2.3	ADO 2.0	RDO 2.0	JET 3.51	VB6

الوصول لأوراكل من وحدات الوصول للبيانات "DAO" وفيجوال بيسك ٣ أصبح ممكنا عند أول مرة أصبح فيها ODBC قابلا للتطبيق، لكن لم يدعم DAO العديد من خصائص أوراكل الخاصة، حتى الوصول للمخططات والحزم كان صعبا، إن لم يكن مستحيلا. إلى حدا ما، ما زال DAO لا يقدم الدعم المطلوب، ولذا تم الانتقال إلى منتجات تدعم قواعد البيانات البعيدة بشكل أفضل. وحدات البيانات البعيدة RDO، التي سيتم شرحها في الفصل ١٦ هي أول طريقة ODBC توفر بعض الخصائص المطلوبة ولكن ليس بالقدر الكافي. ومع ظهور فيجوال بيسك ٦ وADO واستكمال تحسين OO4O، يتوفر وضع أفضل من ذي قبل لمطوري فيجوال بيسك في بيئة أوراكل.

الحاضر

سوف يكون هذا مقتصر على مصادر بيانات ODBC في المقام الأول على الرغم أنه تم إصدار فيجوال بيسك ٦ مع ما يسمى بمزود OLE-DB المحلي لأوراكل مع واحد من خادمي SQL لمايكروسوفت وتعتبر OLD-DB بديل لـ ODBC الشيء الذي سوف تقوم بتغطيته في الفصل ١١ "تقنية جانب المزود".

ومع ADD الذي يأتي مع الإصدار Professional للفيجوال بيسك وكذلك إصدار Enterprise اللذان يزودان بأدوات تصميم قاعدة بيانات أوراكل وعلى الرغم أنهما لا يقدمان دعم كامل إلا أنهما يعتبران البديل بنفس تكلفة الفيجوال بيسك ذاتها أو أكثر. على سبيل المثال، قد

تعمل هذه الأدوات على تكوين جداول بنفس درجة سهولة العمل مع مايكروسوفت أكسس. وكما هو الحال مع أداة GUI أنت بحاجة لفهم ما يحدث بالداخل وإلا لن تكون التصميمات ذات كفاءة.

لقد أصبح الدعم المطور لأوراكل جدير بالملاحظة من خلال فيجوال بيسك ومع الإصدارات السابقة للفيجوال بيسك التي تركت الكثير الذي يمكن لمطور أوراكل استخدامه توجد تطورات الإصدار ٦ بالمميزات التالية:

- دعم OLE-DB المحلي لأوراكل
- دعم مخططات ومترادفات وحزم قاعدة البيانات
- دعم جزئي لحزم
- دعم جداول PL/SQL باستخدام ADO
- مواد قاعدة معرفة أكثر عن أوراكل
- أدوات تصميم قاعدة البيانات

وما زال هناك قليل من مناطق الضعف حيث أن أدوات مصمم قاعدة البيانات لا يمكنها تكوين أو اختيار حزم PL/SQL ولا تعد هذه مشكلة محل اهتمام بالغ، ولكن تعتبر الحزم العلامة الدالة على التصميم ذات الكفاءة. وبطريقة مشابهة فإن طريقة ADO GetSchema لا يمكنها حتى دعم متغيرات خلية PL/SQL.

ونقطة أخيرة بخصوص الموقف الحالي هي أن يقدم الإصدار الثامن لأوراكل هذه الأشياء كأنواع بيانات جديدة وكخصائص ذات علاقة بالوحدة وهذه الأشياء لم يتم دعمها من فيجوال بيسك بعد، حتى مع OO4O.

المستقبل

من الجرأة توقع حدوث أي شيء في عالم تقنية المعلومات، ولكن يعد هذا أحيانا الجزء الأكثر إثارة في العمل مع التقنية الجديدة وفي حيز التنفيذ ليس من المحبب مشاهدة الكثير من التطور في تقنية الوصول إلى البيانات التي يتم فيها تشغيل OLE-DB كخلية قراءة فقط إرسال فقط. وبالنسبة للتطورات التي تشاهدها في الأداء سوف تكون نتيجة مزودين وخدمات وشبكات أسرع وقد تجد أن أساليب تحسين الأداء في طرق الوصول إلى البيانات الأبطأ (مثل الخلايا القابلة للتحديث) قد أصبحت أكثر كفاءة.

ومن حيث الخواص من الممكن أن ننتظر إلى تطورات هامة متعددة. ولا يوجد شك في أن OO4O سوف يمهد الطريق لوجود الميزات التالية به:

- دعم أنواع بيانات الإصدار الثامن لأوراكل LOBs

- دعم الخصائص العلائقية للإصدار الثامن لأوراكل
- القدرة على التسجيل أحداث قاعدة البيانات كبديل من التجميع
- التشغيل غير المتزامن "الذي يشبه التشغيل الموجود في RDO وADO"
- دعم OLE-DB وADO

الانتقال إلى أوراكل

يتم عرض العديد من التطبيقات على سطح المكتب "الشاشة الرئيسية" مثل مايكروسوفت أكسس ويعتبر هذا الحل مكلف مادياً إذا تم تنفيذه بعناية، ومع هذا لا يعتبر أكسس تطبيق مؤسسة فهو لا يحتوي على الخصائص التي تتوقعها (مثل الإجراءات المخزنة) وبالتأكيد لا يقوم بالأداء المطلوب. وقد يكون لديك سوء فهم حول كيفية أداء التطبيق وقد تتوقف عن تطوير النماذج والاستعلامات التي سوف يتم إعادة كتابتها بالكامل. فإذا قررت عمل الإعدادات الأولية بأكسس أو كان لديك تطبيق موجود في أكسس أو خادم SQL فإن القسمان التاليان سوف يعرضا الاختلافات التي عليك توقع حدوثها.

الانتقال من مايكروسوفت أكسس

إذا كنت ما زلت تعمل مع مايكروسوفت أكسس فأنت ما زلت تستخدم النظام حسب اللقب فقاعدة البيانات مخزنة في ملف بامتداد mdb مثل biblio.mdb فأي تطبيق لـه مشغل mdb على الماكينة. يمكنه الوصول إلى الملفات محلياً أو عبر الشبكة ويسمى هذا فنياً خادم المزود ومع هذا فإن الخادم الذي تتعامل معه هو خادم الملف وليس خادم قاعدة بيانات.

وبافتراض أنك قررت أو أوراكل هو الذي يفي بالمطلوب، فعليك مراجعة كيفية الانتقال. وبوجه عام فمن المفروض أن يكون الانتقال سهل نسبياً وما يلي هي بعض النقاط التي يجب عليك وضعها في اعتبارك.

- يحتوي مايكروسوفت أكسس على أكثر من مجرد جداول قاعدة بيانات، فهو يحتوي على تقارير ووحدات رمزية واستعلامات وشاشات ولا يوجد أي من هذه الخصائص متضمنة في أوراكل (أو في أي RDBMS أخرى)، لذا عليك نقل هذه الخصائص إلى ما يقابلهم في فيجوال بيسك.
- وعلى المدى القريب، إذا لم يكن موجود لديك الموارد لإعادة كتابة كل التطبيقات فلديك خيار إرفاق جداول قاعدة بيانات جديدة لأوراكل لقاعدة بيانات أكسس ويتم التشغيل في هذه الطريقة لبعض الوقت وسوف تتم تغطية ذلك في الفصل ١٢ "الوصول لأوراكل من خلال فيجوال بيسك".

- يحتوي أكسس على عمود مواجه لعمل حق مفهرس متزايد آلياً. كما يستخدم أوراكل مولدات أرقام مستقلة تسمى المسلسلات.
- صياغة SQL لأكسس غير متفقة بالمرة مع أوراكل. على سيل المثال، محدد سياق النص في أكسس وفي أوراكل ومحدد جملة SQL في أكسس #.

إذا كنت تريد الانتقال من أكسس إلى أوراكل، استخدم إحدى الطريقتين الموضحتين في الفصل ١٢ "الوصول لأوراكل من خلال فيجوال بيسك".

الانتقال من خادم SQL لمايكروسوفت

يعتبر خادم SQL منتج مختلف تماما من أكسس، على الرغم من أنه متفق تماما مع أكسس. على سبيل المثال، أنواع البيانات. فهو خادم نشط مثل أوراكل ويحتوي على خصائص إضافية جديدة مثل الإجراءات المخزنة والعديد من الوظائف المضمنة ونية متطورة أكثر. ومدة تعلم أوراكل أطول من خادم SQL جزئيا نتيجة أن أوراكل يحتوي على خصائص نهاية طرفية أكبر.

- يعتبر أوراكل RDBMS متعدد الحواسب بينما خادم SQL مقيد بمحطات عمل NT ومعظم أنظمة أوراكل ذات النهاية الطرفية تستخدم UNIX.
- لا يتضمن أوراكل مفهوم قاعدة البيانات الأساسية. فكل قواعد البيانات تعمل بشكل منفرد بملفات البيانات ومعالجة ومراقبة الذاكرة الخاصة بكل واحدة منها.
- بنية أوراكل مختلفة تماما عن خادم SQL وعليك قراءة الفصل ٢ "بنية قاعدة البيانات لأوراكل" ليكون لك إدراك تام حول هذا الاختلاف.
- أوراكل غير متكامل مع معالم ويندوز مثل خادم SQL.
- كلا المنتجين يقوما بتدعيم SQL ويتضمنا إجراءات مخزنة. فبينما خادم SQL يستخدم امتدادات Transact-SQL لـ SQL. فإن أوراكل تستخدم PL/SQL، ووظيفة هذه اللغات متشابهة ولكنهم يختلفوا في الصياغة.
- إجراءات خادم SQL المخزنة تقوم بإرجاع مجموعة السجلات إذا قمت بعمل أمر Select في الإجراءات وأوراكل فقط تعطي تدعم هذا من خلال متغيرات الخلية التي تعد مفهوم صعب قبوله لدى بعض المطورين.
- في الإجراءات المخزنة، يستخدم أوراكل الحركات آليا وفي خادم SQL يتم تثبيت تغييرات البيانات آليا افتراضيا.
- يشترك خادم SQL من حيث مفهوم العمود المتزايد آليا مع أكسس ولكن هذه المرة ممثلة بواسطة خاصية التطابق مع العمود. وفي أوراكل عليك العمل بالمسلسلات.

- خادم SQL يدعم الجداول المؤقتة وهو ليس الحال مع أوراكل.
- في أوراكل لست بحاجة إلى تصحيح مستمر للأخطاء بين المزود والخادم كما تفعل مع خادم SQL.
- قد تختلف الوظائف بين نظامين وبعض الوظائف قد لا يكون لها معادل.
- توجد مجموعة أكبر من أنواع البيانات الأساسية في خادم SQL منها في أوراكل.
- خادم أوراكل لا يدعم خلايا جانب المزود "وهذه الخلايا لابد من تجنبها بأي طريقة".
- يستخدم أوراكل إقفال حسب مستوى الصف. وذلك قبل خادم SQL، بينما خادم SQL يستخدم إقفال حسب مستوى الصفحة.

وبوجه عام يعتبر أوراكل منتج أكثر نضجا وأكثر تناسبا مع بيئات المؤسسة حيث تكون الحاجة إلى ناتج أعلى وإتاحة أكثر واستعادة ونسخ احتياطية يمكن الاعتماد عليها.

ملاحظة لقد أعلن أوراكل عن سلسلة من أدوات الانتقال التي سوف تسهل الانتقال من خادم SQL إلى أوراكل. توجد أدوات تحويل متاحة لنقل قاعدة البيانات ذاتها وكذلك الإجراءات المُخزنة لخادم SQL.

خُلاصة

تعرضنا في هذا الفصل لإمكانات خادم أوراكل وألقينا نظرة على كيفية الفيجوال بيسك ومنتجات مايكروسوفت الأخرى المتعلقة بها. وعلى مدى هذا الكتاب سوف نركز على بنية وتشغيل قاعدة البيانات وتكوين تطبيقات جيدة التصميم وسوف ننتقل إلى عرض كيفية التناسب بين الفيجوال بيسك وبهذا نقوم بتغطية خمس طُرف أساسية للتوسيط مع قاعدة البيانات وإنتاج أكفأ التطبيقات وكذلك استخدام معظم خصائص أوراكل.

٢

بنية قاعدة البيانات

لأوراكل

إذا كنت مستخدم حديث لأوراكل ولا توجد لديك فكرة مسبقة عنه، فسوف تواجـــه بعــض مـــن المفاهيم الغريبة. حتى إذا كنت مطور فيجوال بيسك تستخدم أوراكل للمرة الأولى وتعمل كمحلــل قاعدة بيانات ذو خبرة، فأنت بحاجة ماسة لمعرفة حقيقة قاعدة بيانات أوراكل.

وإذا كنت معتاد على بناء مايكروسوفت أكسس الهيكلي حسب الملف، فـــأنت بحاجـــة أيضـــاً لمعرفة المزيد. وحتى إذا كنت معتاد على قواعد البيانات العلاقية مثل خادم SQL لمايكروســوفت Sybase. فستجد أيضاً التباس في المفاهيم والمصطلحات.

وعلى الرغم من وجود ما هو أكثر لمحلل قاعدة البيانات من محتويات هذا الكتاب، إلا أن فهم البنية سيكون قيماً عند تعاملك مع أوراكل كتطبيق. سوف يزودك هذا الفصل. وفي الحقيقة، قـــد تكون في غفلة تامة عن معرفة مدى إمكانية تراكب مهامك مع محلل قاعدة البيانات على الرغـــم من أن الطريقة لا تتعدى قدر معرفتك لتعليمات البرمجة التي تسبب مشكلة في برنامج.

يتناول هذا الفصل البيانات التي توجد في كل قاعدة بيانات لأوراكل التي يتم توضيحها مـــن خلال الموضوعات التالية:

- الاختلاف بين الطبعة وقاعدة البيانات
- هياكل الذاكرة المطلوب تواجدها في الخادم
- العمليات المستخدمة في تشغيل الطبعة
- ملفات قاعدة البيانات
- إصدارات التوافق والتزامن
- مخططات قاعدة البيانات من وجهة نظر المطور
- نظرة عامة حول كيفية عمل أوراكل

تعتبر كل من هذه المكونات، الذاكرة والعمليات وملفات البيانات مجموعة لا يمكـــن تجزئتـــها ومن الصعب فهم كل واحد منهم بدون الآخرين. وقد تشعر أنه من الأسهل التردد على التفـــاصيل الخاصة بكل مكون منهم على حدة، ولكن الشعور الأفضل أن تجد كل الأشياء في مضمون واحد.

الطبعات وقواعد البيانات

يمكن تخيل قاعدة بيانات أوراكل في طريقتين: كقاعدة بيانات أو كطبعة. فمن الناحية الفنية، تعـــد قاعدة بيانات أوراكل تجميع حقيقي لملفات تخزين البيانات التي توجد في قاعدة البيانات. ولكـــــن من ناحية كونها قاعدة بيانات فلا يمكن للمستخدم التدخل أو التغيير فيها. أما مصطلـــــح الطبعـــة فيطلق على تشغيل قاعدة البيانات. وتتكون الطبعة من هياكل الذاكرة والعمليات المتعلقة بها التـــي تعالج وتتعامل مع هذه الهياكل. وتعد ظاهرة استخدام مصطلحي الطبعة وقاعدة البيانات بالتبـــادل بين العاملين مع RDBMS (نظم معالجة قاعدة البيانات العلائقية لأوراكل) شائعة جـــداً، ويظـــهر

الاختلاف بين الاثنين أوضح عند إدراك أنه من الممكن اتصال طبعات متعددة بمجموعة منفـردة من ملفات قاعدة البيانات الحقيقة، وهذا هو خيار Oracle Parallel Server (الخادم المتـوازي لأوراكل).

عند تشغيل السكريبت لبدء طبعة أوراكل، يتم تخصيص الذاكـرة المطلوبـة وبـدء تشـغيل العمليات وبعد ذلك يقوم بتركيب ملفات البيانات التي تقوم بتكوين قاعدة البيانات. وبهذه الطريقـة يمكن أن يكون لديك أكثر من طبعة قاعدة بيانات على نفس الخادم يمكن تشغيلها بشـكل مسـتقل عن الآخرين. ومع الخادم المتوازي لأوراكل، تقوم أكثر من طبعة بتحميـل نفـس المجموعـة المنفردة من ملفات البيانات. ويقوم كل من الذاكرة وعمليات الخادم وملفـات البيانـات بتكويـن الهياكل التنظيمية الثلاثة الضرورية لفهم أوراكل.

هياكل الذاكرة

الهيكل الأول لأوراكل الذي سوف نقوم بالاطلاع عليه هو فائدة الذاكرة. وتعـد SGA (منطقـة النظام العامة)، وتسمى أحيانا المنطقة العامة المشتركة، مكون الذاكرة الرئيسـي فـي أوراكـل، وأوراكل ما هو إلا قاعدة بيانات على أساس الذاكرة والذي يعني كل البيانات والإقفالات وهكـذا كل المحتويات المتضمنة في الذاكرة، مع الأخذ في الاعتبار ألا يتعدى حجم منطقة النظام العامـة كمية الذاكرة الحقيقية المتاحة على جهاز الخادم، وإلا قد يحدث ترقيم صفحات فعلي في منطقـة النظام العامة، وترقيم الصفحات هذا قد يؤثر على أداء قاعدة البيانات.

ملاحظـة في نظام التشغيل الذي يستخدم نموذج ذاكرة ظـاهري، لا تكفـي أحيانـا الذاكرة الحقيقة المتاحة لكل التطبيقات. وعليه، يقوم النظام بحل هذه المشكلة عن طريق نقل محتويات أقسام الذاكرة، المسماة صفحـات، إلـى قـرص. وتسمى هذه العملية بترقيم الصفحات ويكون حجم الصفحات ٤ ك.بـايت أو ٨ ك. بايت بالتطابق. وقبل تطور أنظمة التشغيل، يتم نقل كل التطبيقات إلى قرص، وتسمى هذه العملية بعملية المبادلة ويبقـى اسـم ملـف القـرص المستخدم كذلك باسم ملف المبادلة.

تحتوي منطقة النظام العامة على المكونات الآتية:

- الذاكرة الوسيطة لقاعدة البيانات
- فاصل أرشيف الإعادة
- منطقة المجمع المشتركة، التي تتكون من المكونات الآتية:
  - ذاكرة المكتبة الوسيطة (التي تتضمن منطقة SQL المشتركة)

- ذاكرة بيانات القاموس الوسيطة (وتعرف أيضاً باسم الذاكرة الوسيطة للصف)

يقوم أوراكل بتخصيص الذاكرة لمنطقة النظام العامة عند بدء تشغيل الطبعة. ويقوم أيضاً بتجميع صفحات ذاكرة مشتركة إضافية حتى يصل حجم ذاكرة النظام العامة القدر المحدد في ملف البدء. يتحكم أوراكل في استخدام الذاكرة ذاتها ولا يقوم بالإصدار حتى يتم إقفال الطبعة، وبالمثل لا يمكن زيادة الذاكرة بشكل فعال. ولتغيير تخصيص الذاكرة، عليك إقفال الطبعة وتغيير معاملات التهيئة وإعادة البدء.

تحذير يعتبر قرار إقفال الطبعة عن قاعدة بيانات الإنتاج قرار حاسم ولا يصح التسرع فيه حيث أنه يتطلب إدراك التأثيرات الناجمة عن العديد من معاملات البدء. وإذا لم يكن لديك خلفية عن إدارة وتشغيل قاعدة بيانات أوراكل. عليك الاتصال بمحلل قاعدة بيانات خبير للمساعدة.

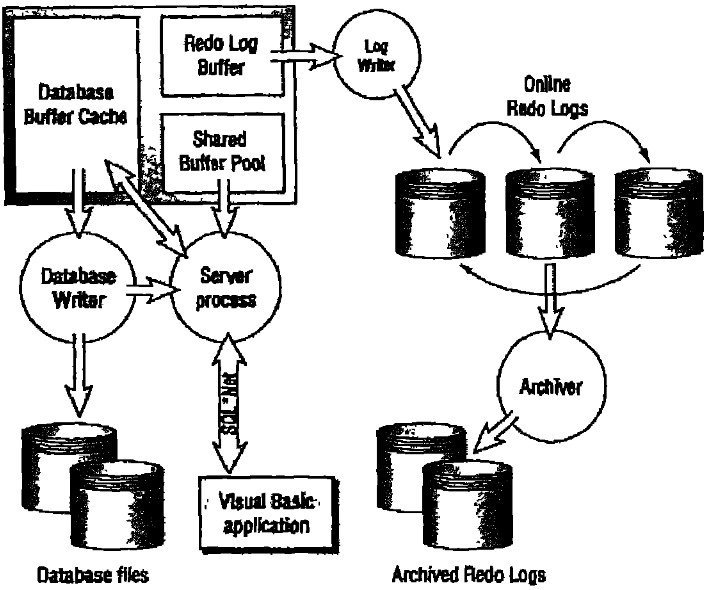
يوضح الشكل ٢-١ هيكل أوراكل بالكامل وتنظيم الذاكرة والعمليات والملفات المستخدمة عموما. وسوف نتناول كل جزء بالتفصيل في هذا الفصل ولكن قد تكون بحاجة للرجوع ثانية إلى هذا الشكل لتذكر نفسك بمكان كل جزء من الآخرين. وفي الشكل سوف تلاحظ منطقة النظام العامة التي تتكون من أجزاء عديدة والتي سوف يتم توضيحها في الأقسام التالية:

الذاكرة الوسيطة لقاعدة البيانات

تعتبر الذاكرة الوسيطة لقاعدة البيانات المكون لمنطقة النظام العامة والتي تقوم بتخزين نسخ من البيانات المقروءة من ملفات البيانات بقاعدة البيانات في الذاكرة والتي تكون جاهزة للوصول إليها بواسطة تطبيق فيجوال بيسك. تتكون الذاكرة الوسيطة من مخازن مؤقتة حجمها الفردي يساوي حجم وحدة قاعدة البيانات في ملفات قاعدة البيانات الحقيقية. تمتلئ الذاكرة الوسيطة هذه حسب طلب التطبيق إذا لم تكن حاضرة فعلا في الذاكرة (راجع الشكل ٢-٢). وعند الاتصال بقاعدة البيانات، يقوم أوراكل بتكوين عملية الخادم (وتعرف أيضا بعملية الظل) التي تعالج متطلبات التطبيق بالنيابة عنك (أو يمكنك الاشتراك في عملية موجودة). وسوف نقوم بتغطية هذه العمليات في قسم "عمليات الخادم" فيما بعد بهذا الفصل.

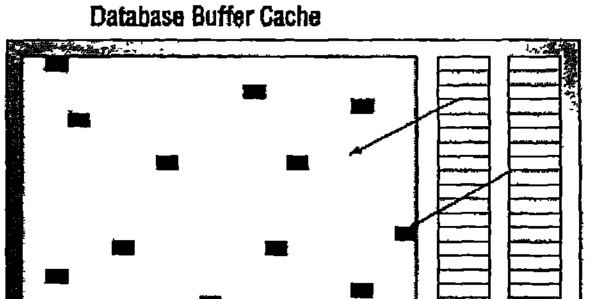
الشكل ٢-١

المكونات الأساسية التي تخص System Global Area ويرمز له بالرمز (SGA).



الشكل ٢-٢

عملية فاصل الذاكرة الوسيطة لقاعدة البيانات.



يوجد قائمتان يقوما بإدارة المخازن المؤقتة في الذاكرة الوسيطة وهما: قائمة the Least Recently Used (LRU) وقائمة Dirty List. وتعد قائمة Least Recently Used (LRU) الأكثر أهمية. ففي كل مرة يتم فيها الوصول إلى الذاكرة الوسيطة بواسطة جملة SELECT مثلاً يتم نقل الذاكرة الوسيطة إلى أعلى القائمة (Most Recently Used). وبالنسبة للذاكرات الوسيطة التي لم يتم الوصول إليها والكائنة أسفل القائمة، فإن مخطط قائمة LRU يؤكد على الوصول إلى الذاكرة الوسيطة الأكثر عموماً وحداثة ويتم إزالة الأقل حداثة. والهدف من LRU الخوارزمية هو تقليل رقم قراءات القرص الحقيقي الذي سيتم إجرائه.

يوجد بأوراكل أيضاً Dirty List تقوم بتتبع أي ذاكرة وسيطة للبيانات تم تغييرها (بمعنى أنه تم تعليمها Dirty List). وعند طلب بيانات لا توجد في الذاكرة الوسيطة لقاعدة البيانات، فإن أوراكل (وعلى وجه التحديد عملية الخادم) يقوم بالبحث في قائمة LRU للذاكرة الوسيطة المجانية. وإذا تم مواجهة ذاكرات وسيطة شائبة على قائمة LRU أثناء هذا البحث، فسوف تقوم عملية الخادم بنقلهم إلى Dirty List. وبعد مسح رقم محدد من الذاكرات الوسيطة، في حالة عدم وجود أي ذاكرة وسيطة مجانية، فسوف تقوم عملية كاتب قاعدة البيانات (Database Writer) بكتابة بعض الذاكرات الوسيطة الشائبة إلى قرص لتحريرهم. وبالنسبة للذاكرات غير الشائبة يمكن نقلها في أي وقت. والذاكرات الموجودة أسفل قائمة LRU هي الأولى بالمشاهدة أولاً.

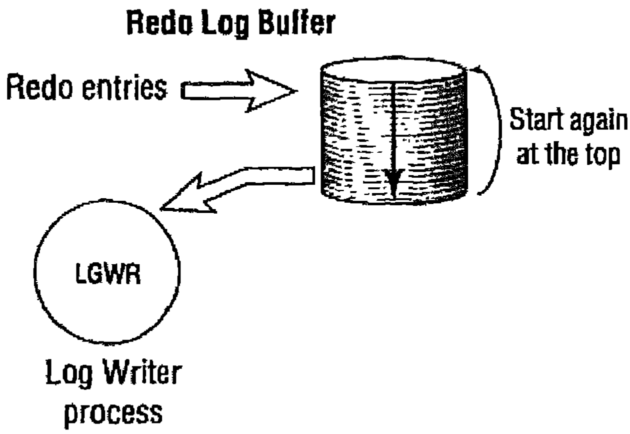
وإذا أمكنك النظر إلى ذاكرة البيانات الوسيطة، سوف تشاهد البيانات الجديدة، وقد يكون قد تم تعديلها وكتابتها إلى ملف البيانات المناسب. وفي الشكل ٢-٢، تلاحظ مناطق عديدة للذاكرة الوسيطة مسودة للدلالة على أنه يوجد بهم شائبة. إذا استخدمت هذا العرض لاستخدام قاعدة البيانات أثناء التنفيذ، فسوف تشاهد الذاكرات الوسيطة تتحول من نظيفة إلى شائبة ثم تعود ثانية. ويظهر هذا غالباً كتأثير واضح والذي يعد واحد من الحالات العامة العديدة التي قد تواجهك عند الاتصال بأوراكل.

الذاكرة الوسيطة لأرشيف الإعادة

الذاكرة الوسيطة لأرشيف الإعادة هي منطقة الذاكرة التي تحتوي على يومية كل التغييرات التي طرأت على قاعدة البيانات؛ وتسمى هذه التغييرات قيود أرشيف الإعادة. فهذه القيود ليس نسخ من قطع البيانات ولكنهم عبارة عن المعلومات المطلوبة لإعادة بناء التغييرات لوحدات البيانات. ولقد تم تجزئة القيود لجعل الحجم صغير بقدر ما يمكن أثناء التحكم الكامل في التغيير. وتستخدم هذه القيود في حالة استعادة انهيار قاعدة البيانات. فلا بد أن نتعامل مع المخازن الفرعية كمنطقة دائرية من الذاكرة، لذلك عندما تصل إلى النهاية الحقيقية من الذاكرة، تبدأ ثانية من البداية. وبالطبع، لابد أن تكتب قيود أرشيف الإعادة على مدار تخزين دائم، وإلا سوف يتم تخطي أرشيف التغييرات، (راجع الشكل ٢-٣). تقوم عملية كاتب الأرشيف "Log Writer" بمراقبة الذاكرة الوسيطة هذه وتسوية محتوياتها إلى مجموعة من الملفات تسمى ملفات أرشيف الإعادة عند اللزوم. تتحكم عمليات الخادم في الكتابة إلى الذاكرة الوسيطة لأرشيف الإعادة ويقوم كاتب الأرشيف بكتابة التغييرات إلى الذاكرة الوسيطة مثل قيود أرشيف الإعادة.

الشكل ٢-٣

عمليـــة Redo Log Buffer



المجمع المشترك

المجمع المشترك عبارة عن منطقة من ذاكرة النظام العامة تحتوي على أجزاء من الذاكرة يمكـن الاشتراك فيها بين المستخدمين. وكما هو موضح في الشكل ٢-٤، فإنه يحتـوي علـى منـاطق عديدة أصغر، كل منها موضح في الأقسام التالية:

الذاكرة الوسيطة للمكتبة

تحتوي الذاكرة الوسيطة للمكتبة على مكونات عديدة يتم الاشتراك فيها بيـن مسـتخدمي قـاعدة البيانات. ويعتبر أهم مركب هو منطقة SQL المشتركة. تحتوي منطقـة SQL المشـتركة علـى التفاصيل حول كل جملة SQL وكذلك خطة التنفيذ (المعروفة أيضاً باسم هيكل التحليل) عند تنفيذ الجملة فعلياً. وبما أنها منطقة مشتركة، فإنها من الممكن أن تحتوي على جمل SQL التي يمكـن استخدامها بواسطة أكثر من مستخدم طالما متطابقان. ولابد أن تكون جمل SQL متطابقة جذريـاً، (في الحالة والمساحة). بالإضافة إلى أن هذه الجمل لابد أن تشير إلى نفس الوحدات. وبسبب هذه المعايير المقيدة، يتم استخدام متغيرات الربط غالباً لجعل الجمل متطابقة جذريـاً. توجـد فـائدة واضحة لهذا المركب وهي تطوير الأداء إذا كان المستخدم يقوم بتنفيذ نفس الجملة التـي يقـوم بتنفيذها شخص آخر لأنه قد تم بالفعل تحليل وإكمال تعليمات البرمجة.

ملاحظـة

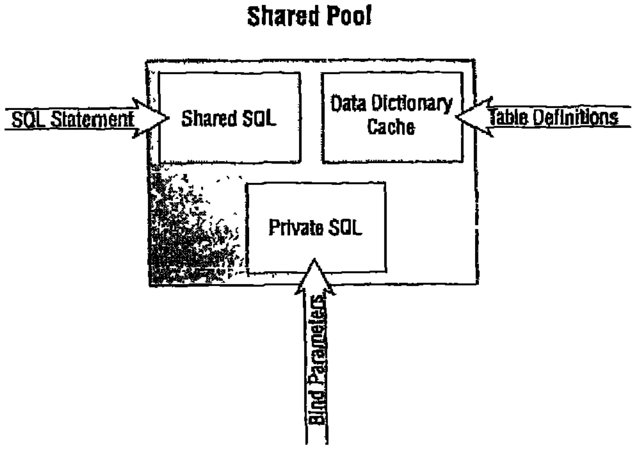
متغيرات الربط عبارة عن معاملات تمر من خلال جملة SQL تسمح للجمل أن تكون متطابقة ما عدا عن قيم المتغيرات الممررة إليهم. وسوف نناقش استخدام متغيرات الربط في الفصل ١٨.

الذاكرة الوسيطة لقاموس البيانات

تقوم الذاكرة الوسيطة لقاموس البيانات في منطقة النظام العامة بتخزين المعلومات حول الجداول الجاري استخدامها. تقوم أيضاً بتخزين أسماء الجداول وأسماء الأعمدة وأنواع البيانات. وبهذه الطريقة تكون المعلومات متناولة فعلياً عند الحاجة إليها لتحليل SQL. إنها متاحة لكل عمليات ومناطق الذاكرة وتسهم أيضاً في التنفيذ كلياً. يمكن تسوية أجزاء الذاكرة الوسيطة لقاموس البيانات بنفس طريقة الذاكرة الوسيطة لقاعدة البيانات (وهذا على أساس LRU).

الشكل ٢-٤

تخطيط المجمع المشترك.



مناطق الذاكرة الوسيطة

يوجد منطقتي ذاكرة جديرتين بالذكر وهما

- المنطقة العامة للبرنامج أو PGA التي تستخدم لاحتواء معلومات المراقبة لعمليات الخادم.
- مناطق الفرز، التي تستخدم للفرز حسب الذاكرة.

العمليات

والآن قد شاهدنا هياكل الذاكرة التي تتطلبها أوراكل وعلينا أن ننظر إلى العمليات المطلوبة لإدارة تلك الذاكرة. تقوم هذه العمليات مع هياكل الذاكرة بتشكيل نواة أوراكل، الطبعة، ويعمل الاثنان جنبا إلى جنب.

تتكون طبعة أوراكل من عدة عمليات في الخلفية يسهم كل منها في قاعدة البيانات بطريقة مختلفة، وهذه العمليات موضحة بإيجاز في الجدول ٢-١. ومن خلال الجدول نجد أن أربع من هذه العمليات إلزامية: كاتب قاعدة البيانات وكاتب الأرشيف ومراقب العملية ومراقب النظام،

وسوف يتم تغطية هذه العمليات بالتفصيل في القسم التالي. وسوف نقوم أيضـا بتغطيـة عمليـة الحافظ لأنك لن تستطيع استعادة قاعدة البيانات كلية بعد فشل القرص بدون التعرف على عمليـة الحافظ هذه. (يمكنك دائما استعادة وقت النسخ الاحتياطي الأخير، ولكن آنذاك قـد تفقـد الكميـة البديلة من البيانات). وأخيرا سوف نقوم بتغطية عمليات المستمع التي تمكن مستخدم الشبكة (مثـل برنامج الفيجوال بيسك) للاتصال بقاعدة البيانات.

الجدول ٢-١: العمليات المختلفة المستخدمة في طبعة أوراكل

اسم العملية	الوصف
الحافظ	تقوم عملية الحافظ بنسخ أرشيف الإعادة إلى القرص أو الشريط لحفظهم.
CKPT	الغرض من CKPT هو تحديث عناوين ملف البيانات بواسطة أرقام تزامـن عند أثناء حدث نقطة الفحص. وتعد هـذه العمليـة اختياريـة، فـإذا لـم تستخدمها، فسوف تتولى عملية كاتب الأرشيف هذه المهمة.
Dnnn	تتحكم عمليات الإرسال في الوصول المشترك إلى قاعدة البيانات لعمليـة أو عمليات User أخرى. يتم تمكين عمليات الإرسال عنـد اسـتخدام خيـار Multithreaded Server. تقوم هذه العمليات باستبدال عمليـات الخـادم وتعالج متطلبات SQL نيابة عن عمليات المستخدم.
DBWR "كـاتب قاعدة البيانات"	يكتب كاتب قاعدة البيانات التي تم تغييرها في الذاكـرة الوسـيطة لقـاعدة البيانات في ملف خارجي ليفسح المجال لبيانات متطلبة حديثة ليتم سـحبها إلى منطقة النظام العامة وهذا هو الغرض منه وليس تثبيت الحركة فحسب.
LCKn "إقفـال العمليات"	يتم تمكين عمليات الإقفال فقط مع خيـار الخـادم المتـوازي (Parallel Server). ويتم استخدام هذه العمليات لحل محتوى الطبعة الداخلي لقطـع البيانات.
LGWR "كـاتب الأرشيف"	يقوم كاتب الأرشيف بنقل البيانات من منطقة النظام العامة إلـى أرشـيف الإعادة المباشرة. يقوم أيضا بتنسيق حدث نقطة الفحـص ويقـوم أيضـا بالتبديل من مجموعة ملف أرشيف إلى مجوعة أخرى.

الجدول ٢-١ : العمليات المختلفة المستخدمة في طبعة أوراكل

اسم العملية	الوصف
PNNN	تستخدم عمليات الاستعلام المتوازية لتوزيع عمل إمداد استعلام قاعدة البيانات.
PMON "مراقبة العملية"	تقوم عملية "مراقبة العملية" بتنفيذ إلغاء تعديلات الحركات في حالة فشل عملية الخادم. وتؤكد عملية "مراقبة العملية" على إقفال موارد قاعدة البيانات بواسطة إصدار العملية الفاشلة.
RECO "عملية الاستعادة"	تقوم عملية الاستعادة بمسح الحركات الموزعة بعد فشل الشبكة.
Snnn "عملية الخادم المشترك"	تستخدم عمليات الخادم المشترك عند خيار Multithreaded Server. تقوم هذه العمليات بمعالجة طلبات SQL المقدمة من عمليات الإرسال.
SMON "مراقبة النظام"	في حالة استعادة الانهيار، تقوم عملية مراقبة النظام باسترجاع قاعدة البيانات منذ آخر حدث نقطة فحص. ويتم تخزين التغييرات في ملفات أرشيف الإعادة المباشرة.
SNP "اللقطات"	تستخدم "عمليات تجديد اللقطات آليا" في تركيب قاعدة البيانات الموزعة لتعميم التغييرات في قاعدة البيانات الرئيسية على قواعد بيانات أخرى. يمكنك استخدامهم أيضا لمهامك الخاصة مثل وظائف الإدارة التكرارية.

وأكثر العمليات أهمية هي DBWR "عملية الكاتب" وLGWR "كاتب الأرشيف" وPMON "مراقبة العملية" وSMON "مراقبة النظام" وفي حالة فشل أي من هذه العمليات، يتم بالتالي فشل طبعة أوراكل. والعملية الاختيارية الوحيدة والتي نوصي بها هي ARCH "عملية الأرشيف" وبدونها لا يتم حفظ البيانات وتصبح احتمالية الاستعادة التامة بعد الفشل ضعيفة جدا. وقد تكون بحاجة للرجوع إلى أحدث نسخ احتياطي كامل وفقد أي تغييرات قمت بها منذ آخر حفظ.

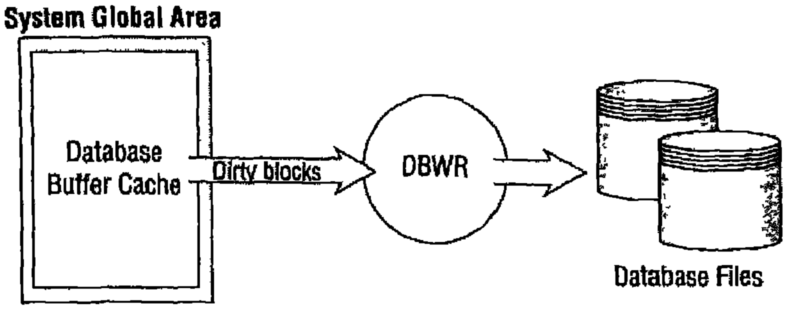
ملاحظة يوجد إصدار مستخدم منفرد لأوراكل به عملية خادم واحدة ولكن اســتخدام هذا يعد نادر جدا. ومن ميزات RDBMS الرئيسية هي ســمات المســتخدم المتعدد ومعظم الطبعات متعددة العمليات كما تم الوصف مسبقا.

تشغيل عمليات أوراكل على ويندوز NT

[illegible] أوراكل على ويندوز NT [illegible] الطريقة [illegible] الأخرى [illegible] UNIX [illegible] Open VMS [illegible] [illegible] في الجدول ٢- [illegible] NT [illegible] Startsid [illegible] أوراكل [illegible] NT [illegible] TNSListener [illegible] SQL Plus [illegible]

كاتب قاعدة البيانات "DBWR"

أي تغييرات تطرأ على البيانات في قاعدة البيانات لابد أن تصنع أولا في الذاكرة. وفيما بعد يقوم أوراكل بتحديث ملفات البيانات: وتتم معالجة هذا بواسطة عمليات كاتب قاعدة البيانات. وهذه هي العملية الوحيدة التي تكتب البيانات من منطقة النظام العامة إلى ملفات البيانـــات. هـــذه العمليـــة موضحة في الشكل ٢-٥. وتقرأ البيانات المقدمة بواسطة المستخدم من ملفات قـــاعدة البيانـــات داخل الذاكرة الوسيطة لقاعدة البيانات إذا لم تكن هذه البيانات فعلا في الذاكرة الوسيطة.



الشكل ٢-٥
تكتب عملية كاتب قاعدة البيانات القطع الشائبة من الذاكرة الوسيطة لقاعدة البيانات إلى ملفات قاعدة البيانات.

أينما تتغير الذاكرات الوسيطة هذه مثلما يحدث مع جمل INSERT أو UPDATE أو DELETE SQL، فإنه يتم تعليمهم "dirty". وعندما يصل رقم الذاكرات الوسيطة في Dirty List إلى قيمة الحد، يقوم كاتب قاعدة البيانات باستخدام قائمة Least Recently Used List للبحث عن الذاكرات الوسيطة المناسبة أكثر لكتابتها إلى الملف لتحرير الذاكرات الوسيطة. ولأن أوراكل يستخدم الهيكل الخاص به، فإن كاتب قاعدة البيانات يمكنه كتابة العديد من القطع في نفس الوقت إلى ملف وبكفاءة (كتابة القطع المتعددة). ومع ذلك، فلأنه لا يكتب البيانات باستمرار، فقد يسبب الانهيار تغييرات في الذاكرة ولا يقوم بحفظهم في ملف البيانات. وتحل هذه المشكلة بواسطة كاتب الأرشيف.

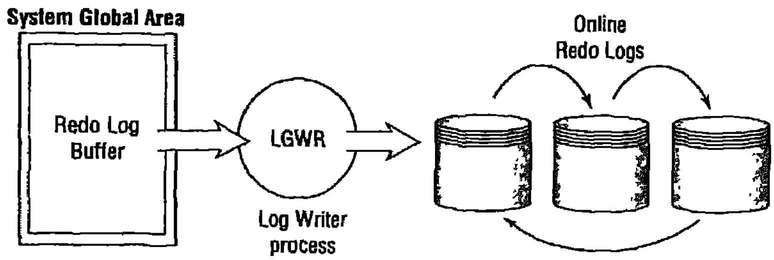
كاتب الأرشيف "LGWR"

أي تغييرات تقوم بها في البيانات في قاعدة البيانات تظهر ليس فقط في الذاكرة الوسيطة لقاعدة البيانات ولكنها تظهر أيضا كقيود داخل الذاكرة الوسيطة لأرشيف الإعادة. في حالة انهيار طبعة أوراكل، يتم افتقاد القيود في الذاكرة الوسيطة. ولتمكين الاستعادة بعد الانهيار، لابد من تخزين القيود في ملفات قرص خارجية. تقوم عملية كاتب الأرشيف بكتابة هذه القيود من الذاكرة الوسيطة لأرشيف الإعادة إلى واحد أو أكثر من ملفات أرشيف الإعادة المباشرة. يوضح الشكل ٢-٦ هذا المحتوى ويجعله أكثر فهما. تذكر أن الذاكرة الوسيطة لأرشيف الإعادة تقوم بتخزين أي تغييرات في البيانات مثل قيود أرشيف الإعادة بحيث يتمكن أوراكل من إعادة إجراء تغييرات في حدث انهيار قاعدة البيانات. وإذا لم تقم بتمكين إجراء نقطة الاختبار، يقوم كاتب الأرشيف بتنفيذ تحديث عنوان ملف البيانات أثناء حدث نقطة الاختبار.

يقوم كاتب الأرشيف بكتابة محتويات الذاكرة الوسيطة لأرشيف الإعادة عند:

- تثبيت حركة قاعدة البينات
- ملء الذاكرة الوسيطة قدر أكثر من الثلث
- تكملة كاتب قاعدة البيانات كتابة الذاكرة الوسيطة إلى القرص أثناء حدث نقطة الاختبار

في حالة عدم حدوث أيٍ من هذه الأحداث، سوف يقوم كاتب الأرشيف بكتابة البيانات دورياً على أية حال كل ثلاث ثوان.



الشكل ٢-٦

تكتب عملية كاتب الأرشيف قيود أرشيف الإعادة من الذاكرة الوسيطة لأرشيف الإعادة إلى الملف.

مراقبة النظام (SMON)

تكمن مهام مراقبة النظام في التالي:

- تنفيذ استعادة الطبعة إذا كان إيقاف قاعدة البيانات السابق لا يحتوي على نقاط اختبار شاملة للنظام. وهذه هي المهمة الأساسية لمراقبة النظام أثناء البدء.
- تكامل الإمتدادات المتجاورة في مساحة حرة في ملفات قاعدة البيانات. ويمكن إيقاف هذا الإجراء في مستوى ملفات الجداول بواسطة إعداد PCTINCREASE = 0 في معاملات التخزين الافتراضية لملفات الجداول.
- المطالبة باسترداد المقاطع المؤقتة الكائنة في ملفات الجداول بعد تحريرهم بواسطة عملية الخادم التي تقوم بتنفيذ عملية الفرز.

مراقبة العملية "PMON"

تقوم عملية مراقبة العملية بمسح منطقة النظام العامة عند فشل عملية الخادم. وعلى وجه الخصوص بإلغاء تعديلات حركة الدورة الفاشلة وإصدار أي إقفالات مورد متعلقة بتلك الحركة.

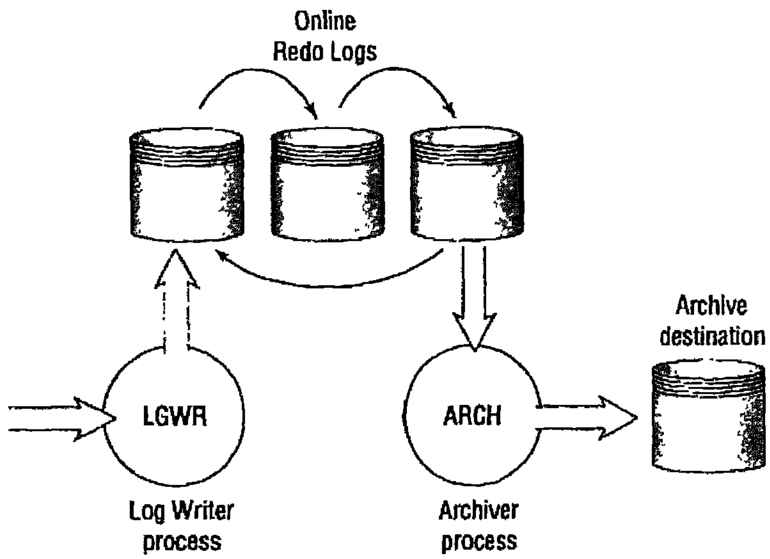
الحافظ "ARCH"

عملية الحافظ ليست إلزامية ولكن موصى بها كثيراً، لأنه بدونها قد لا يمكنك استعادة قاعدة البيانات كلية بعد فشل الوسيط كانهيار القرص مثلاً (راجع الشكل ٢-٧). على الرغم أن عملية كاتب الأرشيف تكتب البيانات من الذاكرة الوسيطة لأرشيف الإعادة إلى قرص، إلا أن ملفات القرص تعد محدودة بالطبع في الرقم والحجم. ويتم معاملة هذه الملفات كمورد معاد، ولذلك عند ملء آخر مورد، يبدأ كاتب الأرشيف ثانية من البداية؛ وآنذاك يصبح الأول مكتوب فوقه، وعليه يتم فقد البيانات. يعرف هذا الإعداد بطور NOARCHIVELOG وهذا هو الافتراض لأوراكل.

ولتحاشي الكتابة الفوقية على ملفات أرشيف الإعادة المباشرة، عليك بدء عملية الحافظ هذه التي تقوم بنسخ كل ملف في منطقة الحفظ مهما أمتلئ الملف. عليك التأكد أيضاً أن قاعدة البيانات في طور ARCHIVELOG. وهذا يعني أنه يوجد خطوتين مختلفتين لتمكين حفظ الآخر الذي ينسى دوماً.ولأن ملفات أرشيف الإعادة المباشرة تستخدم لاستعادة قاعدة البيانات، فإنه يمكنك إدراك أهمية هذه العملية، (سوف تشاهد كيفية ذلك في الفصل ٤ "إدارة قاعدة البيانات").

ويعد الحافظ مسؤول جزئياً عن تحديث ملفات التحكم، مع أن هذه المهمة مع عمليات أخرى مثل عملية كاتب الأرشيف التي تقوم بتحديث ملفات التحكم مع نقطة الفحص ومعلومات مسلسل الأرشيف في تبديل الأرشيف بالإضافة إلى عمليات الخادم التي تقوم بتحديث ملف التحكم عند إضافة ملف الجدول أو إصدار أمر ALTER DATABASE.

تحذير عند ملء وجهة الحفظ وتمكين الحفظ، فلن يكتب أوراكل فوق أرشيف الإعادة المباشرة، ولكنه سوف يتوقف في النهاية.



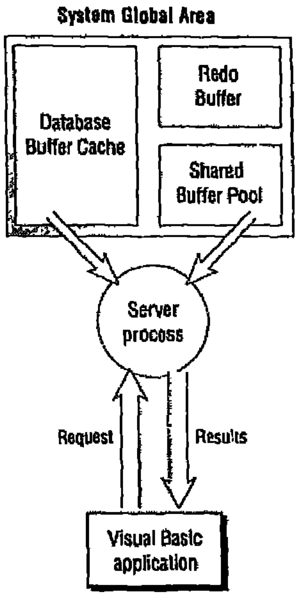
الشكل ٢-٧
تكتب عملية Archiver البيانات من ملفات Online Redo Logs إلى الملفات من الفهرس المطلوب.

عمليات الخادم

تسمى عمليات الخادم بهذا الاسم لأنها العمليات الوحيدة الموجودة على الخادم مصممة خصيصا للتعامل مع طلبات المستخدم، ومع ذلك عادة يلتبس بينها وبين عمليات المستخدم التي تعد التطبيقات الفعلية (مثل فيجوال بيسك). ولكل تطبيق مستخدم، مثل تكوين دورة من برنامج فيجوال بيسك، يتم تكوين عملية الخادم لمعالجة الطلبات. ويعد هذا الإعداد أساسي لكل المستخدمين: لتمكين أوراكل من التدرج إلى مستخدمين أكثر، يمكنك استخدام خيار Multithreaded Server الذي يستخدم عمليات الإرسال وكذلك عمليات الخادم المشترك لمعالجة طلبات المستخدم. يتم التعامل مع تكوين عمليات الخادم هذه في الفصل التالي، ويعطي الشكل ٢-٨ نظرة عامة على كيفية عملهم.

هذه هي المهام الرئيسية لعمليات الخادم:

- تحليل وتنفيذ جمل SQL
- إرجاع مجموعات النتائج إلى المستخدم (برنامج فيجوال بيسك، على سبيل المثال).
- قراءة قطع البيانات إلى الذاكرة الوسيطة للبيانات عند الحاجة إليهم مع كونهم غير موجودين بالذاكرة الوسيطة.
- كتابة تغييرات البيانات إلى الذاكرة الوسيطة لأرشيف الإعادة كقيود أرشيف إعادة.



الشكل ٢-٨: الاتصال بأوراكل من خلال عملية الخادم.

عملية المستمع

عملية المستمع ليست من عمليات الخلفية المطلوبة ولكنها ضرورية إذا كان أي شخص يريد الوصول إلى قاعدة البيانات عن طريق الشبكة (كما هو متعارض مع تشغيل الخادم ذاته). يمكنك بدء تشغيل هذه العملية "لتستمع" إلى طلبات الاتصال القادمة عن طريق الشبكة (تشبه إلى حد كبير مستمع HTTP في خادم ويب). وتقوم بعد ذلك بتكوين عملية الخادم لمعالجة الاتصال بين المستخدم وأوراكل. وسوف نقوم بتغطية هذه العملية أكثر في الفصل ٣ "تكوين قاعدة البيانات".

ملفات قاعدة البيانات

على الرغم أن طبعة أوراكل تتكون من هياكل الذاكرة والعمليات التي تدير تلك الذاكرة، فإن ملفات قاعدة البيانات الحقيقية هي التي تجعل النظام قابل للاستخدام. تقوم هذه الملفات باحتجاز كل البيانات المتعلقة بقاعدة البيانات للأسباب الآتية:

- لتسمح لقاعدة البيانات بالتعامل مع بيانات أكثر من ما يمكن احتجازه في الذاكرة الحقيقية (تذكر أنه لابد من تجنب ترقيم صفحات وتبديل نظام التشغيل).
- لتسمح باستعادة حركات قاعدة البيانات إما إلى نقطة الفشل أو إلى نقطة سابقة في نفس الوقت.

وفي قاعدة البيانات الحقيقية، تعد ملفات قاعدة البيانات هذه ذات أهمية قصوى بالنسبة لعملية النظام القابل للاستخدام. ويوجد أربعة أنواع من ملفات قاعدة البيانات:

- ملفات التحكم
- ملفات البدء أو المعاملات
- ملفات أرشيف الإعادة المباشرة
- ملفات البيانات

كل هذه الملفات تعد هامة جداً عند تشغيل أوراكل، وسوف نراجع كل واحد منهم بدور.

Control Files

ملف التحكم هو عبارة عن ملف صغير يصف هيكل قاعدة البيانات الحالي. ويمكنك أن تتخيله كملف رأس قاعدة البيانات المباشرة.ويعد هذا الملف ثنائي، لذلك لا يمكنك تحريره أو حتى التفكير في ذلك فهو فقط للقراءة بواسطة طبعة أوراكل في البدء ويتم الاحتفاظ بها مفتوحة وفي حالة تحديث حتى يتم إيقاف الطبعة. وتكمن أغراض هذه الملفات في التالي:

- الاحتفاظ باسم قاعدة البيانات.
- تعريف ملفات الأرشيف وملفات قاعدة البيانات.
- تزامن المعلومات المطلوبة للاستعادة متضمنة نقاط الفحص التي بدونها لا يمكنك استعادة قاعدة البيانات مطلقا.

في كل مرة يتغير هيكل قاعدة البيانات للطبعة، عند تكوين أو إسقاط جدول أو ربما إضافة ملف أرشيف، يتم استحداث ملف التحكم عن طريق هذا التغيير.

تحذير افتراض أوراكل هو أن يكون لديك ملف تحكم واحد. وفي حالة تقييم قاعدة البيانات، عليك التأكد من تكوين ملفين تحكم على الأقل. وسوف نقوم بإعداد هذا في الفصل ٣ "تكوين قاعدة البيانات".

ملف المعاملات

يحتوي ملف المعاملات على معلومات حول كل أنواع خيارات البدء لطبعة أوراكل. ويمكن قراءته عند بدء قاعدة البيانات كما هو متشابه في المفهوم لملف البدء وامتداده "*.ini" في

مايكروسوفت ويندوز، يمكنك تحريره بواسطة المحرر المفضل لديك أو إعداد المعـاملات التـي بحاجة إليها يدوياً. وفي حالة تغيير أي إعدادات فلن يتأثروا حتى يتم إعادة بدء الطبعة. يوضـح الشكل ٢–١ ملف معاملات مطابق. وعلى سبيل الافتراض، قد يكـون امتـداده INITSID.ora (حيث يكون SID اسم الطبعة) ولكن سوف يتم إعداد هذا الاسم إذا كـان مولـد عـن طريـق الإمكانات المتاحة. يمكنك أيضاً تغيير الاسم حسب ما تريد، وسوف تتعرف على كيفية اسـتخدامه في الفصل التالي.

تعليمات برمجة ٢–١

```
##########################################################################
# Example INITtest.ORA file
# يحتوي هذا الملف على الإعدادات التي تريدها لطبعة مخصصة لأوراكل.

##########################################################################
#
db_name = test
db_files = 20
control_files = (C:\ORANT\DATABASE\ctl1test.ora,
            D:\ORANT\DATABASE\ctl2test.ora)
Compatible = 7.3.3.0.1
Db_file_multiblock_read_count =  8
Db_block_buffers =  200
Shared_pool_size =  6500000
Log_checkpoint_interval = 10000
Processes =  50
Dml_locks =  100
Log_buffer =  8192
Sequence_cache_entries =  10
Sequence_cache_hash_buckets =  10
Max_dump_file_size = 10240      # limit trace file size to 5 Meg each
Log_archive_start = true       # sets automatic archiving
Oracle_trace_enable = false
# define directories to store trace and alert files
background_dump_dest=%RDBMS73%\trace
```

```
user_dump_dest=%RDBMS73%\trace
db_block_size = 2048
remote_login_passwordfile = shared
text_enable = true
```

Online Redo Log Files

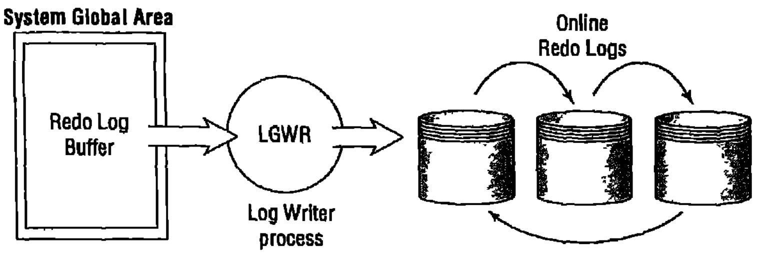
على الرغم من تكوين البيانات في الذاكرة (في الذاكرة الوسيطة لأرشيف الإعادة)، فإنه لابد من وجود نسخ احتياطي في حالة فشل الطبعة. يتحقق هذا عن طريق عملية كاتب الأرشيف التي تكتب البيانات من الذاكرة الوسيطة لأرشيف الإعادة بمنطقة النظام العامة إلى ملفات أرشيف الإعادة المباشرة.يتم التعامل مع ملفات أرشيف الإعادة هذه كمناطق تخزين دائرية (عند ملء الملف الأخير، يبدأ كاتب الأرشيف الدورة من البداية)، وإذا لم تكن قاعدة البيانات في طور ARCHIVELOG، فسوف تتم الكتابة الفوقية تلقائياً.

من الممكن حدوث قليل من التداخل بين مصطلحات ملفات أرشيف الإعادة.ويعتبر افتراض أوراكل لملفين أرشيف؛ ولنسميهما redo_1a ?redo_2a. وتعتبر هذه الملفات أعضاء عن مجموعات مختلفة على الرغم أنه في هذه الحالة يبقى ملف واحد (أو عضو واحد) في كل مجموعة.وفي الحقيقة، من الأفضل جداً وجود مجموعة واحدة أكثر على الأقل لأغراض التنفيذ، لذا اسمح لنا بإضافة مجموعة ثالثة بملف رقم واحد فقط redo_3a. يقوم كاتب الأرشيف بالكتابة إلى redo_1a ?redo_2a ?redo_3a . وبعد ذلك، يبدأ الكتابة إلى redo_1a ثانية بوقت النسخ إلى وجهة الأرشيف. وتسمى المجموعة التي يكتب إليها بالمجموعة الحالية.

ملاحظة لتذكر المصطلح، يكتب كاتب الأرشيف إلى مجموعة أرشيف واحدة ثم إلى الثانية ثم التي تليها.من الممكن أن تحتوي كل مجموعة أرشيف على عضو أو أكثر ولابد أن يكون أعضاء المجموعة الواحدة على أقراص مختلفة. يوضح الشكل ٢-٩ هذا المفهوم جيداً، وقد تكون بحاجة من حين لآخر للرجوع إلى هذا الشكل لمزيد من التذكر.

الشكل ٢-٩

ملفات Redo Log يفضل أن يتم عكسها على البيانات المؤمنة.



للحد من مخاطرة افتقاد ملفات أرشيف الإعادة الهامة هذه التي تعد ضرورية لاستعادة قاعدة البيانات بعد انهيار القرص، اتبع الإرشادات التالية:

- إضافة أكثر من ملف إلى كل مجموعة لتعكس البيانات. يمكنك إضافة redo_1b, redo_2b ?redo_2c كأعضاء جدد للمجموعات الثالثة.
- التأكد من الملفات المنعكسة على أقراص منفصلة بحيث يكون لديك نسخ احتياطي عند فشل أي من هذه الملفات.
- التأكد من تمكين الحفظ
- التأكد أن أراشيف الإعادة على أقراص مختلفة عن أي ملفات بيانات بقاعدة البيانات بما أن الأداء سوف يتأثر من ناحية أخرى.

سوف نقوم بإعداد قاعدة البيانات مع الإرشادات في الفصل ٤ "إدارة قاعدة البيانات"

تلميح

على الرغم أن الانعكاس مع البرنامج أفضل كثيرا من عدمه، إلا أن العديد من محللي قواعد البيانات يوصون بأنه في حالة وجود مربع RAID "Redundant Array of Inexpensive Disks" فإنها تعد أفضل طريقة. ومن الأفضل استخدام مربعين RAID إذا كنت متشككا، ويوصي أوراكل بـ RAID 0+1 بالنسبة لملفات أرشيف الإعادة بينما يعد AID 5 مناسب بالنسبة لملفات البيانات ولكن ليس لملفات أرشيف لإعادة.

ملفات البيانات

ونصل أخيرا إلى ملفات قاعدة البيانات. وهذه هي الملفات الحقيقية حيث يتم تخزين البيانات واعتبارها تمثيل البيانات الحقيقي. وعليك اتخاذ نسخ احتياطي من هذه الملفات بين فترات زمنية محددة.

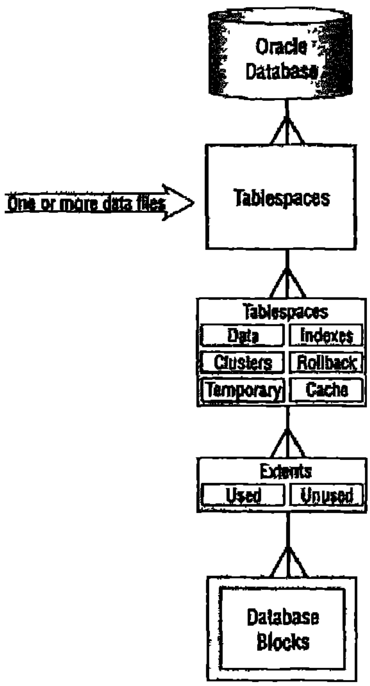
كل ملف من ملفات البيانات، الذي يكونه له امتداد ora أو dbf، يكون مكون في المستوى المنخفض من مجموعة من قطع البيانات المبنية على هيكل ملف نظام التشغيل المحلي. وعلى أساس مستوى القاعدة هذا، يقوم أوراكل بفرض بنيته الاعتبارية.وتعد هذه طريقة من طرق أوراكل التي يحصل عن طريقها على تشغيل قاعدة البيانات على أساس نظم أساسية مختلفة، بينما يكون الاعتماد قليل على نظام التشغيل المحلي.

الهيكل الاعتباري

يقوم أوراكل ببناء هيكل قاعدة البيانات الاعتباري خارج مساحة القرص بحيث تتمكن ملفات البيانات لنظام التشغيل من التزويد له. (تذكر أن مصطلح قاعدة البيانات ينطبق فعلاً على جزء ملف بقاعدة بيانات أوراكل الأمر الذي يجعله مختلف عن الطبعة حسب العملية أو الذاكرة). يحتفظ أوراكل بهيكل اعتباري قائم على ملفات الجداول "Tablespace" ويستخدم الملفات كمورد لكل ملف جدول. ولهيكل موضح رأسياً في ٢-١٠ وتم وصفة في الجدول ٢-٢.

الشكل ٢-١٠

الهيكل الاعتباري لقاعدة بيانات أوراكل.



الجدول ٢-٢: الهيكل الاعتباري لقاعدة بيانات أوراكل

الهيكل	الوصف
قاعدة البيانات (Database)	تعد قاعدة البيانات التجميع الإجمالي للبيانات والتي تعتبر وحدة منفصلة. تكمن قاعدة البيانات حقيقة في كونها سلسلة من ملفات البيانات، وتكمن اعتبارياً في كونها مجموعة من ملفات الجداول.
ملفات الجداول "Tablespaces"	هي عبارة عن وحدة اعتبارية من التخزين يمكن إعدادها بواسطة محلل قاعدة لبيانات لأغراض محددة. يتم طلب وتكوين ملف جدول SYSTEM تلقائياً لاستخدامها في تعريف النظام ومعلومات القاموس. عليك أيضاً إضافة ملف بيانات TEMPORARY كمنطقة عمل مؤقتة للفرز.
	عليك إضافة ملف جداول واحد محدد على الأقل لبيانات المستخدم (وإلا يكون المستخدمون بحاجة إلى الوصول إلى ملف الجداول SYSTEM). وقد تكون الحاجة عادة إلى ملف جداول بيانات واحد على الأقل لكل منطقة تطبيق وآخر للفهارس. وملف الجداول الموصى به أيضاً هو المخصص لاحتواء مقاطع إلغاء التعديلات. وسبب تكوين ملف الجداول الخاص هذا يكمن في أن ملف الجداول قد لا يؤخذ على كونه منفصل إذا كان يحتوي على مقطع إلغاء تعديلات مباشر، ولذلك ينبغي عزل مقاطع إلغاء التعديلات عن جميع المقاطع الأخرى.
	يحتوي كل ملف جداول على واحد أو أكثر من ملفات البيانات الحقيقة والتي لابد من الاحتفاظ بها على نفس القرص إذا لم يوجد الكثير من الأقراص.
	ويتكون ملف الجداول اعتباريا من مجموعة من المقاطع تقوم بتنظيم الهياكل الداخلية. يمكن اعتبار ملفات الجداول بشكل منفصل أو نسخها احتياطيا بشكل مستقل بدون التأثير على باقي قاعدة البيانات؛ ما عدا ملف جداول SYSTEM.
المقاطع "Segments"	يمثل كل مقطع نوع معين في هيكل قاعدة البيانات. ويتكون كل مقطع من مجموعة من الإمدادات.وتوجد أنواع عديدة من المقاطع:
	مقاطع البيانات: يوجد مقطع بيانات واحد لكل جدول غير عنقودي في قاعدة البيانات.
	المقاطع غير العنقودية: يوجد مقطع واحد عنقودي لكل عنقود. وتقوم هذه المقاطع باحتجاز جدول أو أكثر والتي قرر مصمم قاعدة البيانات

الجدول ٢-٢: الهيكل الاعتباري لقاعدة بيانات أوراكل

الهيكل	الوصف
	وضعها معاً لتحسين وقت الوصول عن طريق استعلامات محددة. **مقاطع إلغاء التعديلات**: يوجد دائماً مقطع إلغاء تعديلات واحد على الأقل (الذي يكون في ملف جداول SYSTEM) ويكون أساساً أكثر من ذلك. تستخدم هذه المقاطع لتخزين معلومات عن كيفية إلغاء تعديلات حركة إذا تطلب الأمر ذلك. **مقاطع الفهرس**: تقوم هذه المقاطع بتخزين الفهارس الفردية. **المقاطع المؤقتة**: تستخدم هذه المناطق كمناطق عمل مؤقتة لفرز مجموعات النتائج. توجد هذه المقاطع أساساً في ملف جداول النوع المؤقت، وعليك دوماً تعريف ثم تحديد واحد في أمر CREATE USER. وإذا لم تقم بإجراء ذلك، سوف يخصص أوراكل ملف جداول SYSTEM.
الإمدادات "Extents"	هي عبارة عن مجموعات متلاصقة من قطع البيانات في ملف البيانات الذي يحتفظ بنوع مخصص من المعلومات. تتجمع الإمدادات معاً بشكل اعتباري لتكوين مقطع.
قطع قاعدة البيانات "Database Blocks"	تعد قطع قاعدة البيانات المستوى الأكثر انخفاضا من حيث التخزين الاعتباري في قاعدة البيانات. وتستخدم قطع البيانات لتعريف متطلبات التخزين لوحدات مخطط قاعدة البيانات (مثل الجداول) عند القيام بتكوينهم. وتتمثل كل قطعة قاعدة بيانات في واحدة أو أكثر من قطع نظام التشغيل الحقيقية.

تزامن وتوافق البيانات

والآن قد قمنا بعرض المكونات الرئيسية لقاعدة بيانات وطبعة أوراكل، ولنستعرض الآن منطقة ذات أهمية لكل مطور وهي كيفية قيام أوراكل بتزويد عرض متوافق من البيانات وكيفية إجراء تزامن لهم وخصوصا في المستخدم المتعدد (البيئة الموجهة بالحركة).

وأينما تسمح بوصول أكثر من مستخدم إلى قاعدة البيانات، فمن الطبيعي أن تكون مدرك حول ما يحدث عندما يكون اثنان من المستخدمين بحاجة لتغيير نفس صف بجدول في نفس الوقت وتعد هذه حقيقة في قواعد بيانات أوراكل وAccess وفي أي مكان بينهما. تعد كيفية تعامل

قاعدة البيانات بكفاءة مع مشكلة المستخدم المتعدد أمر بالغ الأهمية بالنسبة للمطور، على الرغم أن حل أوراكل لهذه المشاكل متطور جدا، ونتيجة لذلك يمكنك عدم اعتبارها مشكلة على الإطلاق.

تعالج وظائف أوراكل التالية مشاكل المستخدمين المتعددين:

تزامن البيانات: وهي قدرة قاعدة البيانات على إدارة التحديث لمستخدمين مختلفين لنفس البيانات في نفس الوقت على نظام الإقفال المطور. إذا كنت تقوم بتحديث صف جدول الآن، فإن أوراكل سوف يقوم بإقفال هذا الصف، وعلى المستخدمين الآخرين الانتظار حتى يتم إكمال الحركة.

توافق البيانات: وهي القدرة على استرداد مجموعة نتائج متوافقة عن طريق جملة SELECT، بمعنى أنها لا تتأثر بأي تحديثات أخرى محتملة قد تغير نفس مجموعة البيانات في نفس الوقت. ولابد أن تحتوي أيضا على أي تغييرات قام بها المستخدم ولم يتم تثبيتها بعد. وتسمى هذه الوظيفة غالبا Statement Level Read Consistency.

ملاحظة: تجد معظم مطوري فيجوال بيسك مسرورين جدا بنظام الإقفال الافتراضي. حتى أوراكل يقوم تلقائيا بتوقع وإصلاح حالة عطل الإقفال في حالة وجود اثنان من المستخدمين في حالة انتظار كل منهما للآخر لتكملة التشغيل قبل التمكن من المواصلة. ومع ذلك، يوجد ملحق كامل من الأوامر يتحكم يدويا من إقفال الجدول وبالذات بالنسبة للمطور الذي لديه متطلبات ملحة أو الذي يحاول الحصول على معظم هذه المتطلبات من النظام الخاص بهم. ولابد أن يكون التعامل مع هذه الأوامر بعناية فائقة بالطبع.

إقفال السياسات للتزامن

عندما يريد المستخدم تغيير البيانات، يقوم أوراكل بوضع إقفال مفرد على الصفوف المتأثرة. وبهذا لا يتمكن أي مستخدم آخر من تغيير نفس الصفوف. يضع أوراكل أيضا إقفال مشترك على الجدول. يمنع هذا الإقفال أي مستخدم آخر يقوم بتغيير هيكل أو إسقاط جدول عن طريق جمل Data Definition Language (DDL) (لغة تعريف البيانات).

ملاحظة: تعد قدرة أوراكل على إقفال مستوى صف مساعدة جديرة بالاعتبار من ناحية إمكانيات التحديث.تستخدم معظم قواعد بيانات إقفال مستوى الصفحة الذي يتم فيه إقفال صفوف متعددة على الرغم من وجود صف واحد مستحدث.

تحقيق التوافق

يمكن تحقيق التوافق بطريقة ذكية.عند إعطاء جملة SELECT لأوراكل. يقوم بتسجيل وقت بـــدء الطلب ثم يقوم باسترجاع البيانات. أي صفوف يتم تغييرها بعد بدء جملة SELECT تحتوي علــى بيانات تنتهك خطوط التوافق. يحل أوراكل هذه المشكلة بواسطة كتابة البيانات كما كانت من قبـل التغيير إلى مقطع إلغاء التعديلات. يحتوي مقطع إلغاء التعديلات هـــذا علــى before image (الصورة القبلية) للصف. وعند وصول جملة SELECT إلى الصفوف المتغيرة، يقـــوم أوراكـل بتوجيهها إلى مقطع إلغاء التعديلات لاسترداد المتوافق قبل الصورة. ينطبق هذا الإجــراء علــى جملة SELECT بغض النظر عن حالة التثبيت للصفوف المتغيرة.

ملاحظــة وكما يمكنك الملاحظة، يوجد بأوراكل آلية تزامن وتوافــق متقدمــة جــدا. وتكمن قيمة هذه الميزة في أنه يمكنك كمطور أن تضعها في حسبانك وواثقا أن أوراكل يفعل كل شيء من أجلك.

مخططات قاعدة البيانات

لقد شاهدنا حتى الآن هيكل أوراكل من الداخل والذي يعد محرك قاعدة البيانات الذي يحتوي على العمليات وهياكل الذاكرة وملفات البيانات الأساسية. وقد شاهدنا أيضـــا كيــف تتضمــن ملفــات الجداول الأنواع المختلفة من مقاطع البيانات التي تحتفظ بكل الهياكل الاعتبارية لقاعدة البيانـــات بالإضافة إلى كيفية قيام نظام إقفال أوراكل بمعالجة المستخدمين المتعددين في نفس الوقت.

وهذا هو المنظور الذي ينظر من خلاله محلل قاعدة البيانات دوما. ينظر المطور إلى قـــاعدة البيانات كهيكل مختلف. فقاعدة البيانات عبارة عن نظام يزود المستخدمين بوحدات متعددة لقـاعدة البيانات وسواء كانوا موجودين في المقاطع أو لا فهذا يعد أمر أقل في الأهمية. يبنـــي المطــور منظوره هذا على مخططات قاعدة البيانات حيث يوجد لكل مستخدم في أوراكل مخطـــط قـــاعدة بيانات وهذه هي طريقة تجميع وحدات قاعدة البيانات (مثل الجـــداول والفـــهارس) لذلـــك فـــهم ممتلكون بذلك المستخدم ويمكن حمايتهم من مستخدمين آخرين.

يعد هذا القسم مقدمة إلى عرض قاعدة البيانات، وسوف نشرح قاعدة بيانات الموارد البشـــرية والتعريف بالمهام التي يجب تحقيقها لإعطاء إنتاج قاعدة بيانات جيد. وسوف نستخدم هذه المفاهيم وتطبيقها على مثال ونشاهد كيفية تحقيق كل خطوة وذلك في الفصل ٦ "تنفيذ قاعدة البيانات".

المستخدمون

بدون كلمة سر واسم المستخدم، لن نتمكن من الدخول إلى قاعدة بيانات أوراكل. يتتبع أوراكل كل المستخدمون وكلمات السر المشفرة الخاصة بهم. وعندما تقوم بتكوين مستخدم جديد، يتم بالتالي تكوين مخطط قاعدة بيانات متعلق به ويكون للمستخدم حق الوصول إلى أي وحدات في هذا المخطط، ومع ذلك لابد أن يكون لدى المستخدم الجديد امتيازات محددة قبل السماح له بالوصول مثل القدرة على تكوين دورة في أوراكل وحق استخدام مساحة محددة في ملف الجداول (النصيب).

وبالنسبة لقاعدة بيانات الموارد البشرية، يكون أول إعداد باختبار وإنتاج قاعدة بيانات بمدير قاعدة بيانات واحد أو أكثر وعلى الأقل حساب مستخدم واحد للمطور لكي يمكنه إنتاج تطبيق، ثم يقوم المطور بإعداد حسابات المستخدم لكل موظف يريد الوصول إلى هذه البيانات. ولابد أن يكون مدير النظام حذر جدا لأن البيانات في قاعدة البيانات سوف تكون شخصية أو سرية للغاية. وإلى هذا الحد، من الممكن إعداد جداول محددة لاحتواء المعلومات الهامة والخاصة ووضع قيود تمنع الوصول إليهم عن طريق امتيازات.

الجداول

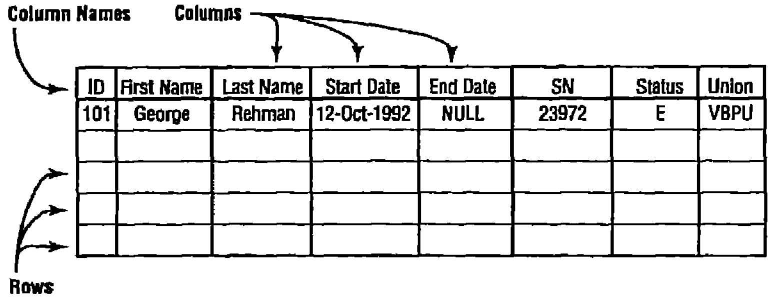
تعد الجداول وحدة التخزين الأساسية للبيانات. تحتوي هذه الجداول على أعمدة وصفوف مثل شبكة جدول البيانات. ويعد هذا تشبيه جيد عند تخيل الجداول وكما هي موضحة في الشكل ٢-١١. ولابد أن يكون لكل جدول في مخطط قاعدة البيانات اسم فريد ولابد أن يكون لكل عمود اسم فريد وبه نوع بيانات متعلق به مثل NUMBER أو DATE.

قد يمتد كل جدول موجود في ملف البيانات إلى واحد أو أكثر من ملفات البياناتextents dba_data_files?. يمكن استخدام عروض _dba لتخصيص امتدادات الجدول.

وعلى الرغم أن أوراكل ٧ يقيد الجداول إلى ملف بيانات منفرد، إلا أن أوراكل ٨ يسمح بتجزئة الجداول والفهارس. وهذا يعني أنه يمكنك أحيانا نشر جدول منفرد على كل ملفات الجداول المتعددة بناءا على سلسلة من القيم في مفتاح. هذه واحدة من الميزات التي تسمح لأوراكل ٨ أن يقوم بمعالجة قواعد بيانات كبيرة للغاية تحتوي على كمية جداول هائلة. يغطي الفصل السادس تكوين الجداول والفهارس بالتفصيل.

وفي قاعدة بيانات الموارد البشرية، يكون المطورين بحاجة إلى تنفيذ تصميم قاعدة البيانات الاعتبارية الخاصة بهم بالإضافة إلى الجداول المطلوبة. وأنواع البيانات التي سوف تكون متاحة هنا هي: بيانات الموظف وسجلات التعليم وبيانات القسم ومعلومات ومحفوظات الرواتب وهكذا.

وكما ذكر آنفا في القسم السابق، لابد من اتخاذ الحذر بحيث تكون البيانات المتضمنة متاحة فقط لهؤلاء الذين لهم حق الوصول إليها.



الشكل ٢-١١. يشبه تخطيط الجدول الجداول الإلكترونية بأعمدتها وصفوفها.

الفهارس

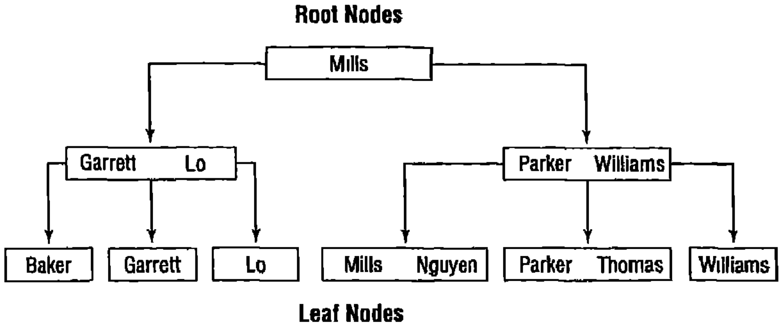
تعد الفهارس وحدة مخطط قاعدة بيانات أخرى مخزنة في المقطع الخاص بها. وأشهر طريقة لتكوين الفهارس هي استخدام الأشجار الثنائية. يستخدم أوراكل بديل مشهور يسمى "B*-tree" لتخزين الفهارس. وهذه الأشجار متساوية بمعنى أن أفرع الشجرة بنفس الطول (راجع الشكل ٢-١٢). وبهذه الطريقة، يتم استغراق نفس الوقت عند البحث عن مؤشر لأي صف من الجدول. وتعد B*-trees سريعة بالنسبة لجمل SELECT وكافية بالنسبة لجمل UPDATE? INSERT ?DELETE.

وكما هو الحال مع أي هيكل شجرة، عند البحث عن صف بيانات محدد، فيكون البدء من القمة (نقطة التوصيل الجذرية) عند العمل من خلال نقاط توصيل الفرع إلى نقاط توصيل الحافة. تحتوي نقاط توصيل الحافة على كل المفاتيح وكذلك ROWID (ROWID? هو المؤشر إلى الصف الفعلي) للصف. وفي كل نقطة توصيل لغير الحافة، يتحدد التوجيه الذي تتبعه بإجراء مقارنة بين المفتاح الذي تبحث عنه والمفاتيح المخزنة في نقطة التوصيل.

يمكنك أن تراجع من الشكل ٢-١٢ كيفية التنقل من نقطة توصيل الجذر إلى نقطة توصيل الحافة. ولنأخذ البحث عن اسم "Garrett" كمثال. وعلينا أولا اختبار اسم البحث أمام نقطة توصيل الجذر "Mills". ويعد اسم البحث هذا أقل من نقطة توصيل الجذر (في سلسلة مقارنة) وبذلك ننتقل إلى نقطة توصيل الفرع التي تحتوي على "Garrett" ?".Lo". وتتم الآن سلسلة المقارن الثانية، ولأن سلسلة البحث ليست أقل من "Garrett" ولا أكبر من ".Lo" فإنها تأخذ المؤشر الأوسط الذي يشير في هذه الحالة إلى نقطة توصيل الطرف الذي نبحث عنه. تحتوي نقطة توصيل الطرف هذه على ROWID الذي يشير مباشرة إلى الصف الذي نبحث عنه في الجدول. وبهذا قد نكون وصلنا هناك إلى ثلاثة طلبات قراءة بدل من مسح الجدول بالكامل.

ملاحظة لقد قدم أوراكل 7.3.3 مفهوم الفهارس النقطية الذي يتم فيه تمثيل كل مفتاح بواسطة إعداد بت في الصورة النقطية.وهذا النوع من الفهارس هام للأعمدة التي يوجد بها قيم مميزة فوق معدل الصفوف ويمكنه أيضا إجراء بحث عن هذه الأعمدة بطريقة أسرع.

الشكل ٢-١٢ مثال مخطط فهرس شجرة B* الذي يستخدمه أوراكل.



وفي قاعدة بيانات الموارد البشرية، قام مصممي قاعدة البيانات بإعطاء دلالة حول كيفية الوصول إلى البيانات.فلابد أن يحتوي كل جدول على فهرس أساسي، وسوف يتم آنذاك تكوين هذه البيانات. وفي حالات عديدة، يدرك المصممون أن الأداء لن يكون مقبول إذا لم يكن لدينا بعض الفهارس الإضافية. وعلى سبيل المثال، قد يكون العمود Last_Name في جدول Employees بحاجة ضرورية إلى فهرس في حالة وصول البيانات بشكل منفرد بواسطة ذلك العمود.

العناقيد "Clusters"

تعد العناقيد طريقة بديلة لتخزين جداول قاعدة البيانات. والفكرة هي أنه يمكنك التحكم في طريقة تخزين البيانات حقيقة بحيث يمكنك استرجاع مجموعات من البيانات بكفاءة بنفس نظام طريقة الإدخال.

عند تكوين مقطع عنقودي، عليك تعريف أعمدة العنقود. وعند تكوين جداول بيانات، عليك تحديد العنقود الذي سوف تضع فيه هذه الجداول؛ وعند إضافة أي جدول إلى العنقود لابد أن تكون به نفس أعمدة العناقيد. ويتم تخزين صفوف الجدول حقيقة في نظام فهرس العنقود، على الرغم أنه من الممكن تخزينهم بواسطة خوارزمية مبعثرة "hashing algorithm".

يمكنك إضافة جداول متعددة إلى العنقود؛ ولكن عليك إجراء ذلك فقط إذا كان يتم استخدامهم معا على الدوام. ومع الجدولين المخزنين في العنقود، لابد أن يكون الربط والمسترجع أسرع لأن الصفوف المرتبطة يتم استرجاعها في نفس قطع البيانات.

وفي قاعدة بيانات الموارد البشرية، قد يقرر المطورين ذلك لأن بيانات الموظف يتم استرجاعها تقريبا مع بيانات القسم، ولابد أن يكون كلا الجدولين في نفس العنقود. وهذه العناقيد بحاجة للمراقبة للتأكد أنهم يزودوا بتحسين أداء شامل.

ملاحظة لا يتم استخدام العناقيد بشكل شائع لأنه يتم إدراك مزايا الأداء عندما تكون الجداول في العنقود ثابتة.وبالنسبة للجداول الخاضعة للغة معالجة البيانات (DML) التكرارية، لا يمكن اعتبارهم مرشحين مناسبين للتخزين في العناقيد.

مولدات المسلسل

مولد المسلسل مدمج في طريقة تكوين أرقام مسلسلة فريدة. والاستخدام الأساسي له هو توفير مفاتيح فريدة للجداول. ويمكن استخدامه بواسطة العديد من المستخدمين بدون أي عروض إضافية وبدون مشاكل إقفال ملحوظة بينما لابد من تكوين الرقم في طرق أخرى.

يتمكن المستخدمون من الوصول إلى المسلسل بواسطة استدعاء الرقم التالي المتاح؛ وعند إجراء ذلك، يمكنك قراءة رقم المسلسل الحالي في أي وقت في نفس الدورة. ولن يكون لدي مستخدمين مختلفين نفس رقم المسلسل من نفس مولد المسلسل (إلا إذا كان المسلسل به تمكين خيار CYCLE) وسوف نناقش هذا بتفصيل اكثر في الفصل ٦.

وفي قاعدة بيانات الموارد البشرية، يقوم المطورين بإعداد مولد مسلسل واحد بنفس الطريقة لكل جدول للتأكد أنه يوجد طريقة منتظمة للتزويد بالمفاتيح الأساسية للجداول.

الإجراءات

يسمح أوراكل بكتابة تعليمات البرمجة إجرائي يمكن تخزينه في قاعدة البيانات. هذه الإجراءات مكتوبة بواسطة PL/SQL، امتداد أوراكل الإجرائي إلى SQL. يسمح أيضاً بكتابة إجراءات ووظائف وزنادات (الزناد هو تعليمات البرمجة التي تعمل تلقائياً عند اتخاذ إجراء محدد إلى الجدول) قاعدة البيانات. كل هذه الأشياء هي وحدات مخطط قاعدة البيانات وهي موضوع الفصل ٩ "مقدمة عن PL/SQL" والفصل ١٠ "تكوين الإجراءات المخزنة".

وعلى مطوري قاعدة بيانات الموارد البشرية اتخاذ قرار حاسم حول قدر التأمين الذي يجب أن تكون عليه قاعدة البيانات. تعطي إجراءات التخزين مثال للإحكام الدقيق على البيانات عن

طريق تقييد ليس فقط هؤلاء الذين قد يكون لديهم حق الوصول إلى أي البيانات ولكـــن لمجـرد معرفتهم طريقة الوصول إليها. ومن الناحية الأخرى، يسمح لك الإجراء بتغليف قوانين الأعمـــال بأسلوب محكم ليس بأسلوب الاعتماد على مطور جانب العميل لكي تتذكر تكامل قوانين الأعمـــال أو، الأسوأ من ذلك، الاعتماد على تكامل الموظفين.

العروض

العرض هو طريقة محددة مسبقا للنظر إلى واحد أو أكثر من الجداول المرتبطة، التـــي تسـمى جداول قاعدة العرض. يتشابه العرض في المفهوم مع الاستعلام في مايكروسوفت أكسس. يمكـــن استخدام العروض لإخفاء المعلومات في جداول القاعدة بنفس طريقة إعطاء المستخدمين وصـــول إلى بعض الأعمدة المتاحة. والميزة أنه يمكن إضافة الأعمدة إلى جداول القاعدة، ولكـــن بعـض التطبيقات القائمة على العروض تستمر في العمل.وتعد هذه العروض نافعة أيضا لإخفـــاء تعقيـد روابط الجدول المتعددة عن المستخدمين.

يمكنك استخدام العروض بنفس طريقة استخدام جداول القاعدة عن طريق أوامر SELECT ? INSERT ?UPDATE ?DELETE علي الرغم من وجود بعض القيـــود التـــي ينبغـــي عليـــك اختبارها قبل الخروج.

تعد "العروض" أداة تحتكم إلى مطوري أوراكل في حالة مثل قاعدة بيانات الموارد البشـــرية. تسمح هذه العروض للمطور بتزويد الوصول إلى أجزاء من الجدول. على سبيل المثال، إذا ظلت معلومات الجدول موجودة في جدول Employee، فمن الممكن إعداد العرض الذي قام بتمثيـــل كل البيانات ماعدا الرواتب. ويكون المطورين ذوات تطبيقات العميل بحاجة إلى التأكيد علـــى أن أدوات التطوير هي التي سمحت لهم بالوصول إلى العروض قبل تثبيت هذه الطريقة. لاحـــظ أن مؤشرات PL/SQL التي سوف نشاهدها في الفصل ١٠ "تكوين الإجـــراءات المخزنـــة" طريقـــة أخرى لتحقيق الغرض من العرض ولكنهم بحاجة إلـــى أداة مثـــل وحـــدات أوراكـــل "Oracle Objects" لـــ OLE (الموضحة في الفصل ١٨ "وحدات أوراكل التي تخص OLE") لكي يمكنـــك الوصول إليهم.

المترادفات "Synonyms"

المترادف هو اسم آخر (أو بديل) للوحدة "object" في أوراكل. يسمح المترادف بتحديد البديـــــل للجداول والعروض والإجراءات والمسلسلات. من الممكن أن تكون المترادفات عامة أو خاصـــة وأن تزود بأمين إخفاء اسم مخطط قاعدة البيانات للوحدة. تعد المترادفات غريبة على أوراكـــل. ومثل مطور VB، لابد أن تكون حذر علي وجه الخصوص عند استخدامهم لأنه لا يوجـــد لـــهم

أساسا دعم تقنية Open Database Connectivity (ODBC) لمايكروسوفت. (سوف نعرض ODBC بتعمق أكثر في الفصل ١١ "تقنية جانب المزود").

ولأن المترادفات تزود بمستوى تأمين إضافي، فإن مطورين قاعدة البيانات العاملين على قاعدة بيانات الموارد البشرية قد يميلوا إلى استخدامهم. ومع فيجوال بيسك ٦ تحسن الدعم لكل المترادفات كثيرا فقد دعمت وحدات أوراكل لـ OLE المترادفات كما نتوقع.

نظرة عامة وكيفية عمل أوراكل

لا يوجد شك في أنه من الصعب احتواء المكونات الداخلية لأوراكل. ففي التطبيق، أنت لست بحاجة إلى الاحتفاظ بكل التفاصيل، ولكن لابد أن تكون على دراية تامة حول ما يحدث. وقد تندهش كثيرا لمعرفة مدى إفادة هذه المعرفة لك حتى لو لم تنتهي من إنهاء تنفيذ مهام محلل قاعدة البيانات. لذلك من الجدير إنهاء هذا الفصل بإجماله في نظرة عامة حول كيفية عمل الدورة النموذجية في أوراكل فعليا.

١- يقوم مدير قاعدة البيانات ببدء الإجراء الذي

- يقرأ ملف بدء المعاملات
- يخصص الذاكرة لمنطقة النظام العامة
- يبدأ تشغيل العمليات المطلوبة
- يفتح ويقرأ ملف التحكم
- يفتح ملفات بيانات قاعدة البيانات للوصول العام

٢- يقوم مسئول قاعدة البيانات ببدء تشغيل عملية المستمع لانتظار طلبات اتصال المستخدم "user-connection".

٣- يعمل مستخدم الفيجوال بيسك اتصال بقاعدة البيانات خلال الشبكة بواسطة استخدام ODBC أو وحدات أوراكل "Oracle Objects" لـ OLE.

٤- تقوم عملية المستمع بإرسال عملية الخادم لمعالجة طلبات SQL للمستخدم (أو قد تكون قادرة على مشاركة عملية الخادم).

٥- يمرر تطبيق الفيجوال بيسك جملة SQL إلى قاعدة البيانات.

٦- يتم تخصيص منطقة من المجمع المشترك لجملة SQL.

٧- يتم سحب البيانات المطلوبة إلى الذاكرة الوسيطة لقاعدة البيانات إذا لم تكن هناك فعلا.

٨- أي بيانات تتغير يتم تكوينها في الذاكرة وتخزينها كقيود أرشيف إعادة في الذاكرة الوسيطة لأرشيف الإعادة.

٩- يرجع التحكم إلى تطبيق الفيجوال بيسك بمجموعة نتائج إذا كان مناسب.

١٠- تكتب عملية كاتب قاعدة البيانات تغييرات البيانات ثانية إلى القرص عند مقابلة معايير محددة.

١١- يكتب كاتب الأرشيف التغييرات إلى ملفات أرشيف الإعادة عند تثبيت التغييرات.

خلاصة

لقد قدم هذا الفصل نظرة عامة عن هياكل الذاكرة الداخلية والعمليات وملفات البيانات لقاعدة بيانات أوراكل. قمنا أيضا بتغطية ملفات البيانات المطلوبة لاحتجاز البيانات الإجمالية في قاعدة البيانات والتي تسمح بالإعداد والاستعادة وقد شاهدنا كذلك نظرة المطور لأوراكل من مخطط قاعدة البيانات. ويظهر بدون غرابة أنه ما ذال المزيد عن أوراكل والكثير منه خلف مجال هذا الكتاب. وسوف نغطي بما فيه الكفاية للتأكيد على أنك، كمطور فيجوال بيسك، تستطيع إدراك المفاهيم جيدا. تتضمن هذه المفاهيم في هياكل الذاكرة التي قام عليها أوراكل وعمليات إدارة تلك الذاكرة.

ولكي تكون محلل قاعدة بيانات جيد، لابد أن يكون عندك دراية أكثر من محتوى هذا الكتاب. وأنصح أنه إذا كنت تريد تطوير هذا الجانب بالمشروع الخاص بك، فلابد أن تأخذ قليل من برامج تدريب وأن تقرأ نصوص مفصلة أكثر. وعلامة محلل قاعدة البيانات الجيد هي كيفية تقسيم نسخة مفاهيم خادم أوراكل "Oracle Server Concepts".

وفي الفصليين التاليين، سوف ننتقل من الجانب النظري إلى الجانب العملي مع إعداد وصيانة قاعدة البيانات لأوراكل.

ملاحظة لمزيد من المعلومات عن أوراكل. راجع إتقان العمل مع أوراكل ٨-١ للكاتب جاتيندر بريم (سيبكس ١٩٩٩).

٣

تكوين قاعدة البيانات

بعد الإطلاع على بنية أوراكل، نحن الآن على استعداد لإنشاء قاعدة البيانات والطبعة الأوليـــان. تعتبر هذه الخطوة عملية متقدمة في بعض كتب الإرشادات الخاصة بأوراكل، ومن الممكـــن أن تصبح مهمة معقدة. في هذا الفصل، سنقوم بإنشاء قاعدة بيانات من البداية إلى النهاية. كمطـــور فيجوال بيسك. نادراً ما تقوم بإنشاء قواعد بيانات، ولكنك ليس أول مطور يقوم بـــدور مسـئول قاعدة البيانات.

عند البدء في إنشاء قواعد بيانات، ستقع في أخطاء كثيرة أو ستتمنى لو لم تفعل بعض الأشياء بهذه الصورة. من المفضل أن تعمل على جهاز للاختبار بحيث يمكنك بناء قاعدة البيانـــات ثـم إعادة بناءها وعمل الأخطاء التي تريدها دون التدخل مع بنية إنتاج. في حالـــة اســتخدام خـــادم ويندوز NT مناسب (يعتبر جهاز بنتيوم ٢، ٢٦٦م هـــ، ٦٤ م.ب. الذي أقوم باســتخدامه جيــداً لقواعد بيانات صغيرة، ولكن إذا بلغت ذاكرة الجهاز ١٢٨ م.ب. فسيكون أفضل)، يمكنك تطويــر قاعدة البيانات وتعليمات فيجوال بيسك البرمجية على نفس الجهاز. لكن مشاركة موارد الذاكـــرة والأقراص على نفس الجهاز ليس إجراءاً نموذجياً من حيث الأداء.

هناك العديد من الخطوات لإعداد قاعدة بيانات. سنقوم بتغطية الموضوعات التالية فـــي هـــذا الفصل:

- اســـتخدام أدوات الإدارة
- تثبيت أوراكـــل
- تثبيت ملفات البدء (تسمى أيضـــاً معـــامل)
- تشغيل وإيقاف قاعدة البيانـــات
- إنشاء قاعدة البيانـــات
- تكوين الشـــبكة

[illegible]

لتشغيل البرنامج المساعد، فمن الممكن أن يصبح هذا الاختلاف بين الأوامر مربكاً إذا لم تكن مستخدماً للحاسب المناسب.

تم إعداد أغلب الأمثلة في هذا الفصل بأوراكل ٨ من أجل ويندوز NT، ولكن الملاحظات المختصرة تعطي أوامر لحاسبات أخرى عندما يكون ذلك مناسباً. على أي حال، تعتبر كيفية تركيب أوراكل من خطوة رئيسية بالنسبة لك لكي تستطيع الفهم. ستجد أنه من الأسهل أن تقوم بإعداد أوراكل على ويندوز NT، كما ستحتاج إلى استشارة التوثيق الخاص بالحاسب بشأن البيئات الأخرى.

أدوات الإدارة

يوفر أوراكل عددا من أدوات الإدارة. في البداية، قد يكون تحديد الأداة التي سيتم استخدامها أمراً محيراً، وللأسف، لا يمكنك استخدام نفس الأداة لكل شيء. مع التعرف على بيئة ويندوز، والتوجيه من قبل موردي قواعد البيانات الآخرين، يحتوي أوراكل على أدوات الوسيط الرسومي لجعل العمل أكثر سهولة. في نفس الوقت، قد لا تفهم ماذا يحدث بالتحديد عندما تجد الوسيط الرسومي يقوم بتنفيذ كل الأوامر المتضمنة.

الأسوأ من ذلك هو أن أدوات الوسيط الرسومي نادراً ما تتيح لك نفس المرونة مثل إصدارات سطر الأوامر. أفضل شيء هو أن تتعلم تقنيات سطر الأوامر ثم تتعامل مع أدوات الوسيط الرسومي عندما ترضى عن هذه العملية. في هذا الفصل، ستقوم بالإطلاع على تقنيات سطر الأوامر ثم تنتهي بالأساليب الرسومية.

تعريف قاعدة البيانات

يمكنك إعداد طبعات أوراكل عديدة كما تريد على الخادم، ويعتبر ذلك مفيداً لفصل المعالجة خاصة عندما تريد إعداد قاعدة بيانات للاختبار. لأنك لا تستطيع إدخال اسم طبعة قاعدة البيانات التي تريد العمل عليها في أدوات الإدارة نفسها، يجب تعريف طبعة قاعدة البيانات الحالية قبل بدء تشغيل الأدوات. بصورة نموذجية، يوجد اسم الطبعة (أو معرف النظام "SID") في متغير بيئة نظام التشغيل. في ويندوز NT، يتم تخزينه في ORCALE_SID. يجب تعيين هذا المتغير قبل أي أداة من الأدوات عن طريق كتابة الأمر التالي:

C:\> SET ORACLE_SID=TEST

[illegible] Test [illegible] NT [illegible]
[illegible] ORACLE_SID [illegible]
[illegible] regedit.exe [illegible] regedt32.exe
\HKEY_LOCAL_MACHINE\SOFTWARE\ORACLE
[illegible]

SQL*Plus

SQL*Plus هو البرنامج المساعد الفعال للمطورين، وهو متاح على كل حاسب يتم تشغيل خادم أوراكل عليه. يفيد SQL*Plus في تكوين وحدات قاعدة البيانات وإعداد المستخدمين ومنح الامتيازات، ولكن لا قيمة لـ SQL*Plus بالنسبة لأي عملية من عمليات قاعدة البيانات الإدارية الأخرى. سنقوم باستخدام SQL*Plus في الفصول اللاحقة بصورة موسعة.

يمكنك بدء SQL*Plus بكتابة C:\> SQL*Plus عند موجه الأوامر أو على حاسب ويندوز، يمكنك بدئها عن طريق النقر المزدوج على رمز التطبيق. أكتب Exit للخروج من SQL*Plus.

ملاحظة لتشغيل سكريبت بواسطة SQL*Plus أو مدير الخادم، الذي يتم توضيحه لاحقاً، قم بكتابة "@" في بداية اسم السكريبت. على سبيل المثال، @create_database.sql.

مدير الخادم

مدير الخادم هو البرنامج المساعد الذي ستقوم باستخدامه في أغلب الأوقات لإعداد وإدارة قاعدة البيانات. يتوفر مدير الخادم على جميع الحاسبات ويحل محل برنامج SQL*DBA. على ويندوز NT، تقوم ببدء البرنامج بكتابة: C:\> SVRMGR30 عند موجه الأوامر الخاص بأوراكل ٨. أكتب Exit للخروج من مدير الخادم.

ملاحظة اكتب SVRMGR23 الخاصة بأوراكل ٧,٣ على NT وSVRMGRL على جميع الحاسبات الخاصة بإصدار طور السطر.

مدير طبعة NT

يكون برنامج طبعة NT المساعد مطلوباً في بيئة ويندوز NT فقط، كما أنك تقوم باستخدامه أساساً لتكوين خدمات NT اللازمة لتشغيل أوراكل على هذا الحاسب. يمكنك أيضاً استخدام هذه الخدمات لتعيين كلمة السر للحساب الداخلي ولتكوين وتشغيل قاعدة بيانات. سنناقش في هذا الكتاب البرنامج المساعد المستخدم لتكوين الخدمات على NT وهذا شيء لا تستطيع عمله أي أداة أخرى.

يمكنك تشغيل مدير طبعة NT بطور سطر الأوامر أو كتطبيق وسيط رسومي. للأسف، لا تعتبر أحدهما مجموعة من الآخر. إنني أفضل استخدام طور الأوامر لتكوين خدمات قاعدة البيانات فقط. تقوم بتشغيل إصدار سطر الأوامر بواسطة أوامر Oradim73 أو Oradim80 الخاصة بأوراكل ٧،٣ وأوراكل ٨ على التوالي. سنقوم في هذا الفصل بعرض تغطية شاملة لمعاملات سطر الأوامر التي يجب استخدامها لتشغيل هذه الأدوات.

ملاحظة

يوجد أيضاً برنامج يسمى Instance Manager (مدير الطبعة). يعد هذا البرنامج تطبيقاً ضمن مدير ضمن مدير المشروع لأوراكل كما أنه برنامج مختلف تماماً عن البرنامج الموضح هنا.

مدير المؤسسة لأوراكل

مدير المؤسسة لأوراكل هو أداة للإدارة الكاملة، وتأتي هذه الأداة مع نسخة إصدارة المؤسسة الخاصة بخادم أوراكل، كما أنه متاح بعيدا عن الإصدارة القياسية. يعتبر مدير المؤسسة لأوراكل أداة مفيدة للمبتدئين تسمح لك بتنفيذ عدد كبير من العمليات. عند تثبيت (تركيب) NT Oracle Enterprise Edition (إصدارة المؤسسة لأوراكل الخاص بـ NT)، يتم تثبيت (تركيب) مدير المؤسسة أيضا. هناك بعض احتياجات الإعداد التي يتم توضيحها بالتفصيل في توثيق مدير المشروع لأوراكل. تأكد من أنك قمت بإعداد ملف جداول للمخزن الذي يحتاجه مدير المؤسسة وقم بتكوين مستخدم خاص يحتوي على ملف الجداول هذا كافتراضي.

عندما تقوم بإعداد مدير المؤسسة، ستكون قادرا على إدارة قواعد بيانات عديدة لأوراكل من نفس الوحدة الطرفية للتحكم. يحتوي مدير المؤسسة لأوراكل على مجموعة متنوعة من الأدوات التي تفيد في مراقبة فاعلية قاعدة البيانات والتحكم في هذه الفاعلية. سنطلع على الكثير بشأن هذه الأداة في الفصل ٤ "إدارة قاعدة البيانات".

على أي حال، يمكنك الاستفادة من حزمة الأداء، وهي إضافة اختيارية لمدير المؤسسة تمدك بنظام مؤهل يقوم بتحليل قاعدة البيانات وإجراء تغييرات موصى بها لتحسين أدائها. بالرغم من

أنك ستأخذ بعض الوقت لمعرفة وإعداد المعلومات التي تحتاجها حزمة الأداء، فسيمكن لحزمة الأداء التوصية ببعض التغييرات التي كانت ستحتاج لسنوات عدة لكي يتم اكتشافها.

بيئة التطوير

ستقوم بيئة التطوير النموذجية بتشغيل ويندوز NT وتثبيت "تركيب" ليس SQL*Plus أو SQL*Net أو Net8 وبرامج تشغيل ODBC الملائمة فقط، بل تقوم أيضا بتثبيت مدير المؤسسة لأوراكل وباني الإجراءات لأوراكل (انظر الفصل ٩ "مقدمة عن PL/SQL"). كما أنك بالطبع ستكون قمت بتشغيل إصدارة المحترفين أو المؤسسة لفيجوال بيسك.

يمكنك تشغيل خادم أوراكل على نفس الجهاز الذي تقوم بالتطوير عليه طالما كنت مستعداً للأداء. إذا فعلت ذلك، إذن حاول الحصول على جهاز ذو سرعة عالية بقدر الإمكان وذاكرة ١٢٨ م.ب على الأقل. سيؤدي ذلك إلى إمكانية استخدام جهاز صغير محمول للتطوير، الذي يفيد خصيصا عندما تكون مسافرا أغلب الأوقات. تعتبر إحدى مميزات أوراكل أنك تستطيع التطوير على الجهاز الصغير المحمول الخاص بك والنشر على خادم يونيكس ذو النهاية الطرفية بمشاكل قليلة. يمكنك النشر لإطار رئيسي أيضا.

البرامج المساعدة الأخرى

بالإضافة إلى برامج أوراكل المساعدة المذكورة مسبقا، هناك عدة برامج مساعدة أخرى هي:

Export and Import (التصدير والاستيراد): تسمح لك هذه البرامج المساعدة بتصدير البيانات المحددة من قاعدة بيانات واستيراد البيانات إلى نفس قاعدة البيانات أو إلى قاعدة بيانات أخرى. سنناقش هذه الميزات في الفصل ٤ "إدارة قاعدة البيانات".

SQL*Loader: يسمح لك هذا البرنامج المساعد استيراد البيانات بكفاءة من ملف نصي، خاصة بالنسبة للتحميل الضخم للبيانات. سنناقش هذا البرنامج المساعد في الفصل ٨ "تحديث قاعدة البيانات".

SQL*DBA: وهو برنامج مساعد تم استخدامه مسبقا لإدارة قاعدة البيانات، ولكن منذ أوراكل ٧,٣ تم إحلاله بمدير الخادم ومدير المؤسسة. لم يعد هذا البرنامج المساعد متضمنا في أوساط تثبيت أوراكل.

تثبيت أوراكل

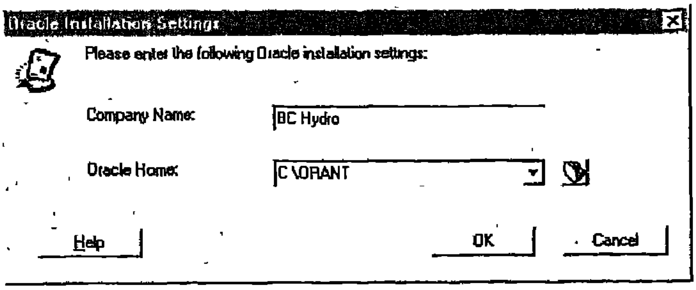
في البداية، لم تكن عملية تثبيت أوراكل أسهل العمليات، ولكن مع أوراكل لـNT، أصبح تثبيته أسهل من العديد من برامج سطح المكتب. بمجرد إدراج القرص المدمج، قم بتشغيل setup.exe وأكتب المعلومات التي يتم مطالبتك بها. بعض الأقراص المدمجة (CD)، مثل إصدار المشروع الخاص بأوراكل ٨، لديها إمكانية التشغيل التلقائي لبرنامج الإعداد. يسمح لك هذا الإصدار بتثبيت نموذج لقاعدة بيانات (قاعدة بيانات اختبار)، يفيد ذلك كثيراً حيث أنك تستطيع بذلك مشاهدة العملية أثناء إجرائها. بالرغم من أن تكوين قاعدة بيانات أوراكل يمكن أن يكون مزعجاً في بادئ الأمر، إلا أن بإمكانك استخدام قاعدة بيانات عرض لكل الأعمال التي تحتاج للقيام بها.

تلميح يعد حجم قاعدة بيانات الاختبار (٣٠ م.ب) صغيرا جدا لأي تطبيقات واقعية، كما يتم ضبطه ببساطة. عند الحاجة إلى توفير متطلبات اختبار واقعية، يجب أن تكون قاعدة البيانات أكبر وأكثر فعالية. عند الضرورة، يمكنك طلب المساعدة من مسئول قاعدة بيانات ذو خبرة.

كبديل لقاعدة بيانات الاختبار، يمكنك في أوراكل ٨ استخدام مساعد قاعدة البيانات لأوراكل وهو معالج يطالبك بكل المعلومات المناسبة. (لا تزال بحاجة إلى معرفة ما سوف تقوم بعمله، على أي حال). لا تهتم لغة فيجوال بيسك بنوع قاعدة البيانات، ويستطيع نموذج قاعدة البيانات العمل بسرعة أكبر. يتم تثبيت أوراكل نموذجيا في دليل ORANT على ويندوز NT. في هذا الدليل، ستجد كل البرامج الخاصة بأوراكل بالإضافة إلى البرنامج المساعد وملفات نموذج قاعدة البيانات. عند تشغيل برنامج التثبيت، يجب إدخال اسم الشركة الخاصة بك والدليل الأساسي كما هو موضح في الشكل ٣-١. نحن نفترض هنا أنك تختار القيمة الافتراضية. يجب أيضا تحديد المنتجات التي سيتم تثبيتها (انظر الشكل ٣-٢). ستحتاج لمنتجات الخادم فقط عند هذه النقطة. أخيرا، ستحتاج إلى تحديد ما إذا كنت تريد نموذج قاعدة بيانات أم لا. دائما ما يكون نموذج قاعدة البيانات جديرا بالاهتمام إذا لم تكن قد قمت بتكوين قاعدة بيانات أوراكل من قبل لأنه يعطيك شيئا تعمل منه وتضاهيه.

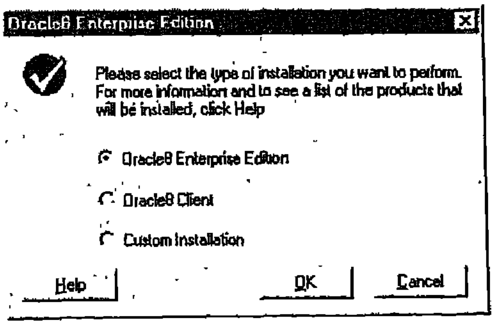
الشكل ٣-١

تحديد المكان الذي سيتم تثبيت أوراكل فيه.



الشكل ٣-٢

تحديد منتج أوراكل المقرر تثبيته.



أوراكل على الحاسبات التي تعتمد على الترتيب الأبجدي

حاليا، تعتبر الحاسبات من نوع Alpha based رائدة الأداء طالما كنا مهتمين بأجهزة NT، يجب فقط مشاهدة شاشة التوقف ثلاثية الأبعاد لترى الفرق. تقوم Alphas بتكوين خادمين شديدي الفعالية؛ لكن كيف يكون حاسب NT عمليا؟

يوجد إصدار خاص لأوراكل للأجهزة من نوع Alpha-based. بالإضافة إلى ذلك، يتوفر مضاه يسمى FX32 والذي يسمح لك بتشغيل تعليمات برمجية من نوع Intel-based ٢، وقد تلقى هذا المضاهي رأيا مؤيدا. على أي حال، يجب أن تكون حذرا بشأن تنفيذ هذا النوع من البرامج.

يمكنك الوصول في النهاية إلى دليلي نظام. اكتشفت أنا من تجربتي الشخصية كم يكون الإتلاف التام لدليل نظام NT سهلا، عن طريق تثبيت بعض برامج تشغيل ODBC في إحدى الحالات. (لا ينتج أوراكل أيا منهما لا ألفا على وجه الخصوص). لا يعد هذا عيبا في NT قد يحدث في أي حاسب آخر، فالأمر هو مجرد أن هناك العديد من خيارات البرامج للحاسب.

عند تثبيت أوراكل على NT، أوصي بأن لا تقوم بتشغيل تطبيقات أخرى على الخادم حيث أنها تستهلك ذاكرة ضرورية، خاصة التطبيقات التي تعتمد على الوسيط الرسومي. أيضاً، لا تقم بتشغيل مضاه لأن هذا قد يسبب بعض المشاكل. قم بتخصص الخادم لأوراكل واحتفظ به هكذا.

الخدمات المتاحة على NT

عند تثبيت أوراكل، تقوم بتثبيت برامج تنفيذي (أو صور) تقوم بتشغيل العمليات المختلفة التي رأيناها في الفصل ٢ "بنية قاعدة البيانات لأوراكل". مع ويندوز NT، يتم تشغيل هذه العمليات كسلاسل منفصلة لخدمة NT. في حالة استخدامك لبنية قاعدة بيانات أوراكل، سيتم إعداد كل الخدمات المطلوبة لك. في حالة عدم استخدامك له، ستحتاج لتثبيت خدمتين على الأقل، وتقوم بإجراء ذلك بواسطة مدير NT. الأمر الذي تقوم باستخدامه هو:

```
C:> ORADIM80 -NEW -SID test -INTPWD password -STARTMODE AUTO
-PFILE c:\orant\database\inittest.ora
```

دعنا نفحص هذا السطر عن قرب:

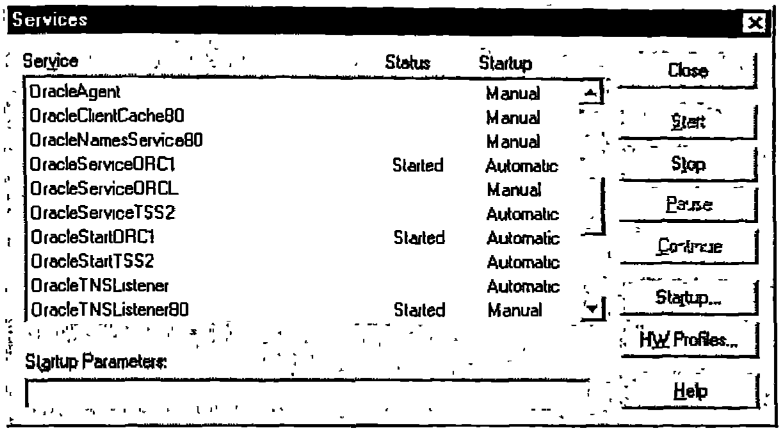
- NEW يحدد أن هذه خدمة جديدة.
- SID هو معرف النظام الذي يعرف الطبعة الخاصة (في هذه الحالة نقوم بتعيين SID إلى "TEST").
- INTPWD هي كلمة السر لحساب INTERNAL. يجب إدخال كلمة السر هذه دائماً. مع أوراكل ٨، لا يمكنك الاتصال كـ INTERNAL دون ملف الكلمة السر.
- STAT MODE يحدد ما إذا كنت تريد بدء قاعدة البيانات تلقائياً "AUTO" أو يدوياً "MANUAL" عند بدء تشغيل النظام.
- PFILE هو ملف البدء الذي قمت بإعداده بالفعل لطبعة قاعدة البيانات هذه (سنتناول ذلك في القسم التالي).

بتنفيذ أمر مدير طبعة NT بنجاح، فإنك الآن قد قمت بتكوين خدمتين جديدتين لقاعدة بيانات تسمى "test"، الخدمة الأولى هي "OracleServiceTest" التي تقوم ببدء عملية ORACLE80.exe لقاعدة البيانات. الخدمة الثانية هي "OracleStartTest" التي تضمن بدء قاعدة البيانات تلقائياً في حالة إعادة تشغيل الخادم. بهذه الطريقة، يستطيع أوراكل الاستمرار في تشغيل بالرغم من عدم تسجيل المستخدم بالدخول إلى الخادم.

ملاحظة في حالة اختيار معامل STARTMODE اليدوي في أمر Oradim80، لا يتم إذن تكوين خدمة Oracle Start Test.

الآن، يجب مراجعة الخدمات التي قمت بتكوينها. يظهر ذلك تحت لوحة تحكم الخدمات التي قمت بتكوينها. سترى شيئا مشابها لما في الشاشة المعروضة في الشكل ٣-٣.

الشكل ٣-٣ لوحة التحكم في الخدمات.



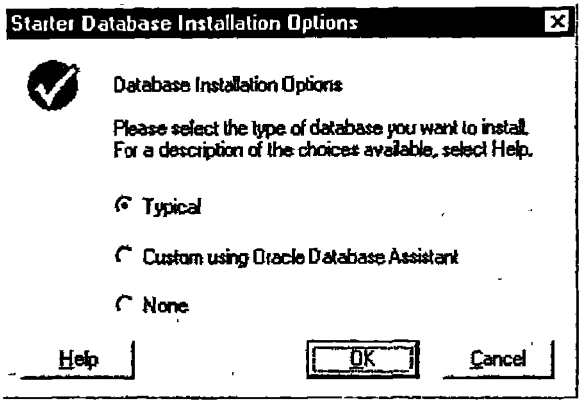
ملاحظة على حاسبات غير ويندوز NT، ستجد أن عليك إجراء تغييرات في إجراءات بدء تشغيل النظام لضمان أن الطبعة تم تشغيلها عند قيامك بإعادة التشغيل. استشر مسئول النظام لمعرفة التفاصيل الخاصة بكيفية القيام بذلك.

مساعد قاعدة بيانات أوراكل

مع أوراكل ٨ لويندوز NT، لديك الآن الخيار لاستخدام معالج يسمى Oracle Database Assistant أو مساعد قاعدة بيانات أوراكل الذي سيساعدك خلال خطوات تكوين قاعدة البيانات. سترى مربع الحوار الموضح في الشكل ٣-٤ والذي يمكنك اختيار تثبيت قياس أو تكوين قاعدة بيانات مخصصة. تحتوي حاسبات أخرى، من بينها ٧,٣ على NT، على نماذج قواعد بيانات وليس المعالج.

الشكل ٣-٤

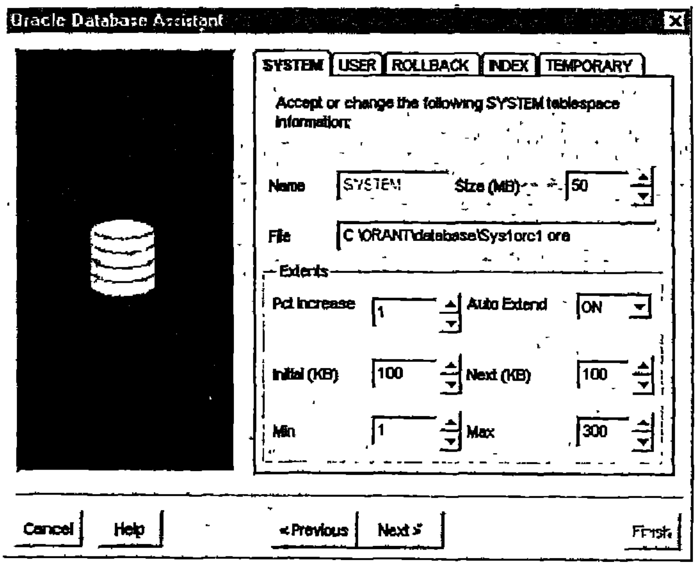
تحديد نوع التثبيت.



لبدء قاعدة بيانات وتشغيلها بسرعة، فإن خيار Typical هو أسهل خيار ويوفر لك نموذج قاعدة بيانات يسمى orcl. إذا أردت شيئا أكثر تحديدا، فسيعرض لك الشكل ٣-٥ إحدى الإطارات العديدة التي ستراها أثناء العمل مع المعالج. سيقوم المعالج بتكوين كل الملفات التي تحتاجها. يمكنك تفصيل هذه الملفات بتاريخ لاحق.

الشكل ٣-٥

إطار نموذجي لنافذة مساعد قاعدة بيانات أوراكل.



يجذب أسلوب الوسيط الرسومي مستخدمي أوراكل الجدد لأنه يوفر افتراضات عديدة ويقوم بتكوين الملفات المناسبة. لا يمكن القيام بوظائف الإعداد العديدة دون التدخل اليدوي، ولا يمكنك اتخاذ قرارات ذكية دون معرفة الكثير عن ما تقوم بعمله. يعتبر اتخاذ هذه القرارات الذكية هو موضوع باقي هذا الفصل.

ملفات البدء

بمجرد تثبيت أوراكل وأي خدمات مطلوبة على الخادم، تحتاج إلى أن تقوم بتخطيط خصائص قاعدة البيانات. لتكوين قاعدة بيانات أوراكل، يجب أولا تحديد كل المعاملات التي تحتاج إليها. يستخدم أوراكل مفهوما مألوفا لكل مطوري ويندوز: وهو ملف نصي يحتوي على معاملات بدء. يمكنك إدخال عدد كبير من المعاملات، كما هو الحال مع أي ملف بدء، ولكننا سنلتزم بالمعاملات الأكثر أهمية. يستخدم ملف البدء هذا أينما تقوم ببدء طبعة. عند تغيير المعاملات، يجب إيقاف الطبعة وإعادة تشغيلها لتنفيذ تغييراتك.

مع أي تثبيت لأوراكل، ستجد نموذج ملف يسمى init.ora وهو يعد نقطة بداية للملفات الخاصة بك. (على NT، انظر في دليل c:\orant\database) يمكنك تحرير هذا الملف وتسميته الاسم الذي تريده ولكن تأكد من أنك لم تقم بتوثيقه في أي مكان. أفضل طريقة هي تضمين اسم الطبعة في اسم الملف. على سبيل المثال، بالنسبة لقاعدة بيانات تسمى "Test"، قم بتسمية ملف البدء inittest.ora.

ملاحظة عندما يتم إصدار أمر STARTUP في مدير الخادم دون خيار PFILE يكون الاسم والموقع المفترض لملف المعامل هو %ORACLE_HOME%\database\init%ORACLE_SID%.ora.

تحتاج الآن إلى تحرير ملف البدء. إذا قمت بمراجعة الملف النموذج، فسترى عدة معاملات يوجد لها ثلاثة خيارات تشمل قواعد البيانات الصغيرة والمتوسطة والكبيرة الحجم. يتم التعليق على جميع القيم فيما عدا قيم قواعد البيانات الصغيرة بعلامة "#".

تلميح

عندما يطلق أوراكل على قاعدة بيانات أنها "صغيرة"، تكون قاعدة البيانات هذه ذات حجم متوسط. إن أوراكل مصمم لمعالجة كميات ضخمة من البيانات (حتى تيرابايت). يمكنك استخدام إرشادات الأحجام التالية التي تأتي ضمن أوراكل ٨ لـ NT: بالنسبة لأنظمة دعم القرار الموجهة للقراءة، تكون قاعدة البيانات الصغيرة أقل من ١٠ ميجابايت والمتوسطة أقل ١٠٠ ميجابايت والكبيرة تكون أكبر من ١٠٠ ميجابايت. أما بالنسبة لنظام معالجة المعاملات المباشر الموجهة للكتابة، تكون قاعدة البيانات الصغيرة حتى عشرة مستخدمين متزامنين والمتوسطة تكون حتى ١٠٠ مستخدم متزامن والكبيرة تكون أكثر من ١٠٠ مستخدم متزامن. يجب أن يستخدم أغلب المستخدمين أنواع قواعد البيانات المتوسطة والكبيرة. عند الشروع في استخدام قاعدة بيانات كبيرة جدا، فقد حان الوقت للحصول على بعض المساعدات للضبط عالي الجودة.

هناك العديد من المعاملات الممكنة التي يمكنك وضعها في ملف بدء. على أي حال، من الأفضل الالتزام بالمعاملات الموضحة في الجداول ٣-١ إلى أن تصبح أكثر تمرسا وخبرة.

الجدول ٣-١ : معاملات البدء المقترحة لقاعدة بيانات صغيرة

المعامل	الوصف
Background_dump_dest	موقع ملفات التتبع التي تم توليدها بواسطة الإجراءات الخلفية.
db_name	اسم قاعدة البيانات، وهو معامل إلزامي يصل عدد حروفه إلى ثمانية أحرف.
db_files	أقصى عدد للملفات التي تستطيع قاعدة البيانات فتحها.
control_files	يحدد هذا المعامل عدد ملفات التحكم كلها وأسمائها. يجب تحديد أثنين على الأقل.
Compatible	إصدار الخادم الذي يجب أن تتوافق قاعدة البيانات هذه معه.

الجدول ٣-١: معاملات البدء المقترحة لقاعدة بيانات صغيرة

المعامل	الوصف
Db_block_size	حجم قطع قاعدة بيانات أوراكل بالبايت. يمكن تعيين هذا المعامل عند تكوين قاعدة البيانات فقط. القيمة الافتراضية هي ٢٠٤٨، ولكن بالنسبة لأكثر من قاعدة بيانات صغيرة جدا أو عندما تكون لديك أعمدة واسعة، يجب تكوين عدد مضاعف لهذا الرقم، مثل ٤٠٩٦، لقاعدة بيانات ذات معاملات مكثفة. بالنسبة لدعم القرار، يوصي أوراكل بأكبر قيمة مسموحة لـ db_block_size. يعقد هذا الحد الأقصى على الحاسب ولكنه سيكون ٨١٩٢ على الأقل.
db_block_buffers	عدد القطع المخزونة مؤقتا في الذاكرة الوسيطة لقاعدة البيانات. قم بضرب هذا المعامل بـ db_block_size للحصول على حجم الذاكرة الوسيطة.
db_file_multiblock_read_count	عدد القطع التي تمت قراءتها في عملية إدخال وإخراج واحدة. يعتبر ٨ عددا معقولا للعمليات ذات الحركات المكثفة و١٦ عددا معقولا لدعم القرار.
Shared_pool_size	حجم مجمع الذاكرة المشترك الذي يتضمن الذاكرة لوسيطة للمكتبية والذاكرة الوسيطة لقاموس البيانات. القيمة الافتراضية هي ٣,٥٠٠٠,٠٠٠.
User_dump_dest	موقع ملفات التتبع التي تقوم عمليات الخادم بتكوينها.
log_buffer	حجم الذاكرة الوسيطة لأرشيف الإعادة في SGA بالبايت.
log_archive_start	يقوم هذا المعامل بتمكين الحفظ التلقائي (أي يقوم ببدء إجراء الخلفية ARCH) إذا تم تعيينه إلى صواب.
log_archive_dest	الدليل الذي توجد فيه ملفات الأرشيف المحفوظ.

الجدول ٣-١ : معاملات البدء المقترحة لقاعدة بيانات صغيرة

المعامل	الوصف
log_archive_format	تنسيق اسم الملف الخاص بملفات الأرشيف المحفوظ.
Rollback_segments	أسماء مقاطع إلغاء التعديلات ستصبح مباشرة عند بدء تشغيل قاعدة البيانات.
oracle_trace_enable	المعامل الذي يتحكم فيما إذا كان البرنامج المساعد Oracle Trace (التتبع لأوراكل) ممكنا لقاعدة البيانات. قم بتعيين ذلك إلى False على NT ولن يتم استخدام Oracle Trace على قاعدة البيانات.

ملاحظة تعتبر القيم المقترحة المعطاة لملف معامل البدء في الجدول ٣-١ مجرد مقترحات. عندما تصبح قاعدة البيانات أكبر ويصبح الأداء أكثر أهمية، يجب استثمار بعض الوقت في الاعتياد على تغيير المعاملات لكي تتناسب مع قاعدة بيانات الخاصة بك. تسمى هذه العملية بعملية الضبط ومن الممكن أن تكون معقدة جدا. لا يجب أن تضع في اعتبارك حجم قاعدة البيانات فقط بل ونشاطها أيضا، سواء كانت معالجة مباشرة ودعم القرار.

يوجد مثال لملف بدء موضحا في القائمة ٣-١. قد تم اشتقاق أغلب هذه المعاملات من المقترحات الموجودة في ملف البدء الافتراضي "init.ora". إذا قمت باستخدام نموذج قاعدة البيانات أو مساعد قاعدة بيانات أوراكل، سيتم إذن تكوين ملف مماثل. يمكنك استخدامه كأساس لإجراء تغييرات لضبط قاعدة البيانات.

تعليمات برمجية ٣-١

```
###########################################################################
# Example INITtest.ORA file
#
يتضمن هذا الملف الإعدادات التي تريدها لطبعة خاصة لأوراكل.
###########################################################################
db_name = test
db_files = 100
control_files = (C:\ORANT\DATABASE\ctl1test.ora,
            D:\ORANT\DATABASE\ctl2test.ora)
compatible = 8.0.3.0.0
db_file_multiblock_read_count =  8
db_block_buffers =  200
db_block_size = 4096

shared_pool_size =  6500000
log_checkpoint_interval = 10000
processes =  50
dml_locks =  100
log_buffer =  8192
sort_area_size = 65536
sequence_cache_entries =  10
sequence_cache_hash_buckets =  10
max_dump_file_size = 10240        # تحديد حجم ملف التتبع ليكون كل ملف ٥
ميجا.
log_archive_start = صواب # تعيين الحفظ التلقائي.
log_archive_dest = %RDBMS80_ARCHIVE%
oracle_trace_enable = خطأ # تحديد الدلائل لتخزين ملفات التتبع والتنبيه.
background_dump_dest=%RDBMS80%\trace
user_dump_dest=%RDBMS80%\trace
remote_login_passwordfile = shared مشترك
text_enable = true صواب
```

ملاحظة تقرأ المعاملات التي توجد بين "%" من السجل في ويندوز NT أو من متغيرات البيئة التي يتم تعريفها عند مستوى النظام أو المستخدم.

تشغيل وإغلاق قاعدة البيانات

مع معاملات قاعدة البيانات المعرفة في ملف البدء، يجب بعد ذلك أن نعرف كيفية بـدء تشـغيل وإغلاق قاعدة بيانات. إذا لم تقم بتثبيت قاعدة بيانات العرض، فلن يكون لديـك أي شـيء تبـدأ تشغيله قاعدة بيانات. إذا لم تكن قد قمت بتثبيت قاعدة بيانات العرض، فليس لديك أي شيء تبـدأ تشغيله حتى الآن. على أي حال، لتكوين قاعدة بيانات، لا تزال في حاجة إلى أمر Startup لأننا نحتاج إلى البدء بطبعة قبل أن تقوم فعلياً بتكوين قاعدة البيانات الجديدة.

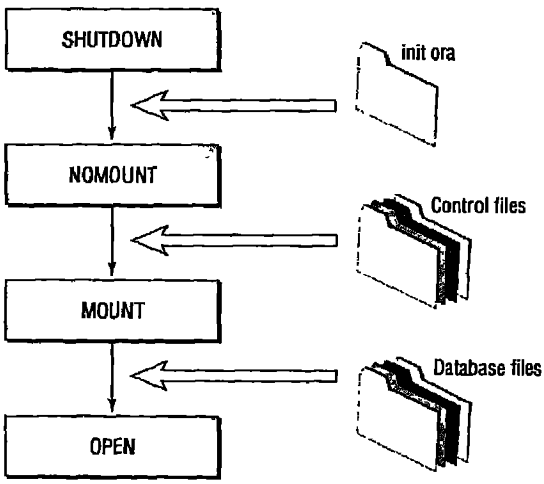
يمكن أن تكون طبعة قاعدة البيانات في حالة من أربع حالات عند تشغيلها، وتمر الطبعة بكـل هذه الحالات (انظر الشكل ٣-٦). هذه الحالات هي:

SHUTDOWN: مغلقة تماما، تسمى Idle (عاطلة عن العمل).

NOMOUNT: بدء العمليات وتخصيص SGA ولكن لا يتم فتح أي ملف

MOUNT: فتح ملفات التحكم

OPEN: فتح كافة الملفات والإمكانية الكاملة للوصول إلى قاعدة البيانات.



الشكل ٣-٦
الحالات الممكنـة لقاعدة بيانـــات أوراكل.

بدء تشغيل قاعدة بيانات

لبدء تشغيل قاعدة بيانات، نستخدم أداة Server Manager "مدير الخادم" والتي يتم إحضارها من موجه سطر الأوامر (انظر الشكل ٣-٧). في ويندوز NT، سـيكون SVRMGR30. قبـل بـدء وإيقاف قاعدة بيانات، يجب الاتصال بقاعدة البيانات من خلال أوامر CONNECT INTERNAL

أو CONNECT/AS SYSDBA. تقوم باستخدام أمر STARTUP بالحالة التي تريد أن تكون فيها قاعدة البيانات. على سبيل المثال، إذا أردت فتح قاعدة بيانات موجودة بالفعل لتوفير إمكانية وصول عامة، تكتب:

STARTUP OPEN PFILE=c:\orant\database\initdemo.ora

لبدء الطابعة المعطلة عن العمل قبل تكوين قاعدة بيانات جديدة، تكتب

STARTUP NOMOUNT PFILE=c:\orant\database\inittest.ora

يقوم معامل PFLE المذكور في المثالين بتعريف ملف البدء الذي تريد استخدامه لقاعدة البيانات. أما إذا تركت المعامل PFLE، فسيبحث أوراكل عن ملف المعامل المسمى %ORACLE_SID%.ora في الدليل %ORACLE_HOME%\database. هناك قواعد ممثلة لأوراكل على Unix وOpen VMS.

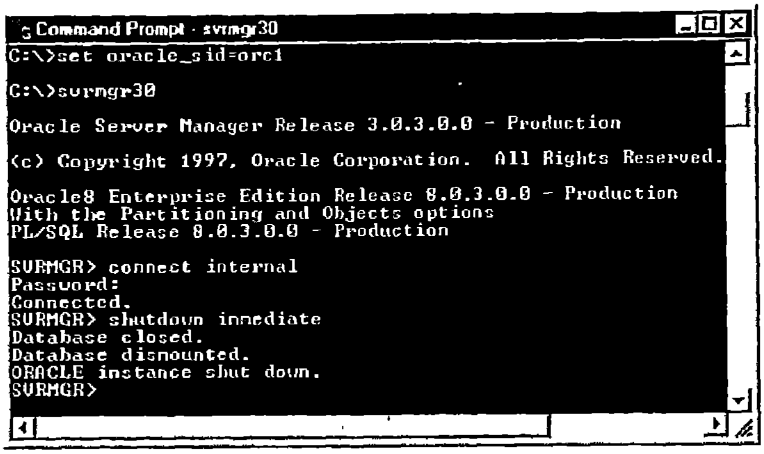
ملاحظة في Unix وopen VMS، يتم إحضار طور السطر الخاص بـ Server Manager عن طريق svrmgrl. يقوم أوراكل ٧,٣ لـ NT باستخدام svrmgr23.

يمكنك فقط استخدام أمر startup عندما تكون قاعدة البيانات مغلقة. أما إذا كانت قاعدة البيانات في حالة متوسطة "NOMOUNT أو MOUNT"، إذا فسيكون عليك استخدام أمر ALTER DATABASE لترقية قاعدة البيانات إلى الحالة التالية، أي من NOMOUNT إلى MOUNT أو من MOUNT إلى Open. على سبيل المثال، يمكنك استخدام

ALTER DATABASE OPEN

التي تقوم بترقية قاعدة البيانات من حالة Mount إلى حالة Open.

تلميح يجب الاحتفاظ بتوثيق دقيق عن ملف البدء الذي تستخدمه، لأنه في حالة وجود عدة ملفات بدء على القرص، قد يؤدي ذلك إلى التباس شديد.



الشكل ٣-٧

بدء تشغيل طبعة قاعدة البيانات.

إغلاق قاعدة بيانات

لإغلاق قاعدة بيانات يتم استخدام أمر SHUTDOWN في Server Manager (انظر الشكل ٣-٨) لديك ثلاث خيارات:

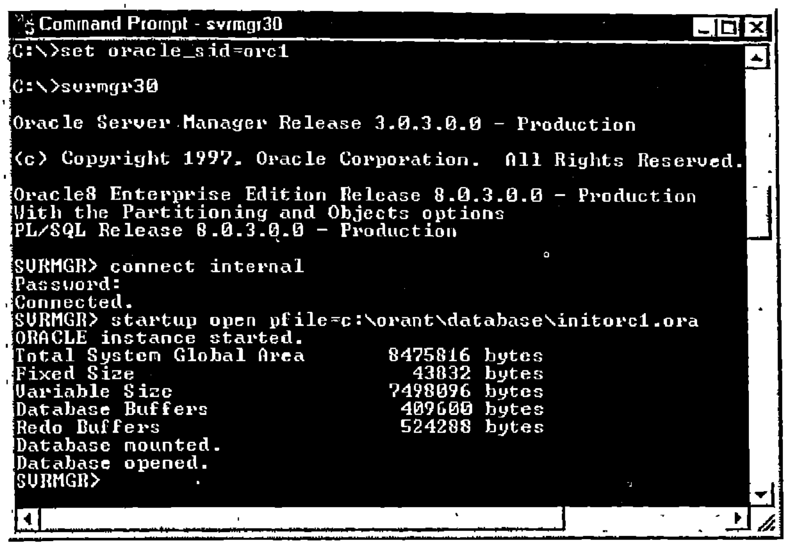
SHUTDOWN NORMAL: يقوم هذا الخيار بإيقاف قاعدة البيانات عندما يكون كافة المستخدمين غير متصلين، ولا يسمح بأي اتصالات جديدة في الوقت الحالي.

SHUTDOWN IMMEDIATE: يقوم هذا الخيار بقطع اتصال كافة المستخدمين على الفور وإلغاء تعديلات أي حركات جارية، ثم يقوم بإيقاف قاعدة البيانات.

SHUTDOWN ABORT: يقوم هذا الخيار بإيقاف قاعدة البيانات على الفور دون إلغاء تعديلات أي حركات. يؤدي استخدام خيار الإيقاف هذا إلى ترك قاعدة البيانات في حالة ما بحيث سيتطلب إصلاح العطل عند التشغيل التالي. يجب استخدام هذا الخيار عند الطوارئ فقط.

تلميح

يعد تخزين أوامر STARTUP وSHUTDOWN في ملفات سكريبت قصيرة أمرا مفيدا. بالرغم من أن طول الأوامر يبلغ سطرا واحدا، إلا أن كتابة ملف سكريبت يعني أنك ليس بحاجة إلى تذكر أو كتابة اسم ملف البدء، الأمر الذي يؤدي إلى الإزعاج عند الاستخدام المتكرر. كما يعد ذلك أسلوبا أفضل لتوثيق النظام الخاص بك.



```
Command Prompt - svrmgr30
C:\>set oracle_sid=orcl

C:\>svrmgr30

Oracle Server Manager Release 3.0.3.0.0 - Production

(c) Copyright 1997, Oracle Corporation.  All Rights Reserved.

Oracle8 Enterprise Edition Release 8.0.3.0.0 - Production
With the Partitioning and Objects options
PL/SQL Release 8.0.3.0.0 - Production

SVRMGR> connect internal
Password:
Connected.
SVRMGR> startup open pfile=c:\orant\database\initorcl.ora
ORACLE instance started.
Total System Global Area        8475816 bytes
Fixed Size                        43832 bytes
Variable Size                   7498096 bytes
Database Buffers                 409600 bytes
Redo Buffers                     524288 bytes
Database mounted.
Database opened.
SVRMGR>
```

الشكل ٣-٨

إيقاف طبعة قاعدة بيانات.

تكوين قاعدة البيانات

بعد أن تعرفت على كيفية بدء الطبعة، يمكنك الانتقال إلى تكوين قاعدة بيانات جديدة. هناك عدة طرق للقيام بذلك. سنتناول هنا أكثر الطرق مرونة، وسنقوم بتكوين وظيفة SQL فورية لتقوم بهذه المهمة. بهذه الطريقة، سيكون لدينا سجل بكل ما قمنا بعمله، الأمر الذي تكون له فائدة كبيرة إذا احتجنا في أي وقت لنسخ قاعدة البيانات إلى خادم آخر أو إذا اضطررنا لإعادة تكوينها بعد حدوث مشكلة في محرك الأقراص.

أمر SQL الذي نستخدمه هو CREATE DATABASE. وما يجب الانتباه له جيدا هو عبارات هذا الأمر، حيث أن بمجرد تكوين قاعدة البيانات، لا يمكن تغيير بعض المعاملات. يتم توضيح أكثر هذه المعاملات استخداما في الجدول ٣-٢. ستصبح معاملات أخرى أكثر أهمية عند تكوين قاعدة بيانات كبيرة جدا، ولكن تعتبر المعاملات الافتراضية كافية جدا حتى الآن.

ملاحظة

سنتناول SQL بمزيد من التفاصيل في الفصل ٦ "تنفيذ قاعدة البيانات"، حيث نقوم بتكوين جداول قاعدة بياناتك، وفي الفصول التالية حيث نقوم بتكوين استعلامات وتحديث قاعدة البيانات باستخدام SQL.

الجدول ٣-٢: معاملات CREATE DATABASE الأكثر أهمية

المعامل	الوصف
LOGFILE	يسمح لك هذا المعامل بتعريف ملفات أرشيف الإعادة المباشرة. يجب تحديد مجموعتين على الأقل، كما يجب أن يكون لديك عضوية في كل مجموعة. من المفيد إبراز العضو ورقم المجموعة (مثال log1a.rdo للعضو "a" الخاص بالمجموعة الأولى وlog2a.rdo للعضو "a" الخاص بالمجموعة الثانية).
MAXDATAFILES	أقصى عدد لملفات البيانات التي يمكن فتحها بواسطة طابعة قاعدة البيانات.
DATAFILE	يقوم هذا المعامل بتحديد الملفات التي يتم استخدامها لملف جداول النظام. إذا حذفت هذا المعامل، سيتم تكوين ملف بيانات في الدليل %ORACLE_HOME%\database.
ARCHIVELOG	يحدد هذا المعامل أنك تريد تمكين الحفظ. (الافتراض هو عدم الحفظ). إذا قمت بتحديد ذلك، يجب أيضا بدء عملية الحفظ بعد إنشاء قاعدة البيانات، لذلك لا تستخدم هذه العبارة في جعله CREATE DATABASE.
CHARACTER SET	تقوم هذه العبارة بتحديد مجموعة أحرف قاعدة البيانات. في حالة حذفها، يكون الافتراض عبارة عن مجموعة أحرف ASCII مكونة من ٧ بت وتسمى "US7ASCII " ولكن أوراكل يوصي بأن تقوم باستخدام "WE8ISO8859P1 " لا يمكنك تغيير هذا الافتراض دون إعادة تكوين قاعدة البيانات.

ستقوم بتكوين قاعدة بيانات تسمى"Test" كمثال، للقيام بذلك، سنكتب أمر CREATE DATABASE في ملف نصي يسمى create_test.sql كما هو في القائمة ٣-٢.

تعليمات برمجية ٣-٢

```
CREATE DATABASE test
  LOGFILE
    GROUP 1 ('c:\orant\log1a.log', 'd:\orant\log1b.log') SIZE 100K,
    GROUP 2 ('c:\orant\log2a.log', 'd:\orant\log2b.log') SIZE 100K
  MAXDATAFILES 20
  DATAFILE 'c:\orant\system.dbs' SIZE 10M
  CHARACTER SET WE8ISO8859P1;
```

سيقوم الأمر أعلاه بتكوين قاعدة بيانات ذات مجموعتين من ملفات الأرشيف، لكل مجموعة عضوان. يمكن فتح ملفات بحد أقصى ٢٠ ملف بواسطة هذه الطبعة. يحتوي ملف جداول النظام على ملف بيانات محدد، وتحتوي قاعدة البيانات على مجموعة أحرف تتكون من ٨ بت. يقوم أيضا الأمر بتعطيل تمكين الحفظ عند تكوينه. سنقوم بتمكين الحفظ لقاعدة البيانات هذه في الفصل التالي.

ملاحظة يعتبر ملف جداول النظام قلب قاعدة البيانات، ويتم استخدامه لاحتواء قاموس البيانات. يقوم أيضا ملف جداول النظام باحتواء أي زنادات ووظائف وإجراءات مخزنة. يجب ألا تقوم نهائيا بتكوين تطبيقات أو أي وحدات تطبيقات أخرى في ملف الجداول هذا.

الآن يمكنك تشغيل سكريبت في القائمة ٣-٢ لتكوين قاعدة بيانات، ولكن من الأفضل والأكثر فائدة أن يكون لدينا سكريبت واحد يقوم بعمل كل شيء. لا يزال يوجد لدينا مزيد من وحدات قاعدة البيانات التي نريد أن تقوم بتكوينها. نحتاج بعد ذلك لتشغيل وحدتي سكريبت واللتان يوفرهما أوراكل. السكريبت الأول catalog.sql، ويقوم بتكوين قاموس البيانات في ملف جداول النظام. السكريبت الثاني هو catproc.sql، ويقوم بتثبيت الوحدات التي تحتاجها PL/SQL (لغة أوراكل الإجرائية)، يتم إضافة كل منهما لسكريبت create_test.sql كالآتي:

```
@c:\orant\rdbms80\admin\catalog.sql
@c:\orant\rdbms80\admin\catproc.sql
```

ستعتمد مواقع وحدات سكريبت هذه على حاسبك وإصدار برنامج خادم أوراكل. إذا كانت لديك الفرصة، فهناك فكرة جيدة هي إدخال أي أوامر تكوين ملف جداول آخر في ملف السكريبت

هذا. سنتناول ملفات جداول التطبيقات في الفصل ٦ "تنفيذ قاعدة البيانات". إذا كنت تعرفها الآن، يمكنك إدخالها في السكريبت.

هناك أيضا وحدتان أخريان يمكنك إضافتهما للسكريبت: rollback segments (أي مقاطع إلغاء التعديلات) وtemporary tablespace (أي ملف الجداول المؤقت)، وسيتم مناقشتها في الأقسام التالية.

مقاطع إلغاء التعديلات

مقاطع إلغاء التعديلات (مجموعات من الإمتدادات) هي ما يقوم أوراكل باستخدامه لتخزين نسخ before-image للوحدات المتغيرة بحيث يمكن إلغاء تعديلات الحركة. عند تكوين قاعدة بيانات، يتم تكوين ملف جداول System ويحتوي تلقائيا على مقطع إلغاء تعديلات نظام. يمكن لمقطع إلغاء التعديلات أن ينمو ليحتوي على امتدادات عديدة كالمحددة في معامل MAXEXTENTS للمقطع. عندما تقوم بتنفيذ ملفات جداول تطبيقات (وهذا ما سيتم أغلب الوقت)، يجب أيضا تكوين مقطعي إلغاء تعديلات تطبيقات أو أكثر. يحتاج خادم أوراكل إلى تكوين مقطع إلغاء تعديلات واحد على الأقل خارج ملف جداول SYSTEM بحيث يمكن ظهور لغة معالجة البيانات "DML" على ملفات جداول التطبيقات هذه. إذا قمت بتكوين مقاطع خاصة، إذن فلا بد من تعريفها في ملف البدء. نقوم بتكوين مقاطع خاصة في ملفات جداول منفصلة كما يلي:

```
CREATE TABLESPACE rollback01
  DATAFILE 'C:\ORANT\DATABASE\ROLLBACK01.ORA' SIZE 500K REUSE
     AUTOEXTEND ON NEXT 500K MAXSIZE 2M ;
CREATE ROLLBACK SEGMENT r1 TABLESPACE rollback01
     STORAGE (INITIAL 50K NEXT 50K MINEXTENTS 2) ;
```

ملاحظة يسمح لك أوراكل بتكوين مقاطع إلغاء تعديلات عامة، ولكن يوصي بذلك للتكوين الخاص لخادم المتوازي لأوراكل فقط.

يمكن الاستفادة بمقاطع إلغاء التعديلات في قاعدة بيانات مكثفة التحديث، وستوجد ملفات الجداول على أقراص منفصلة لا يستخدمها أي شخص آخر. القاعدة هي وجود مقطع إلغاء تعديلات واحد لكل أربع حركات متزامنة.

ملفات الجداول المؤقتة

عندما تقوم بفرز البيانات في قاعدة البيانات، سيحاول أوراكل إجراء ذلك في الذاكرة. في حالة عدم وجود مساحة كافية من الذاكرة لإجراء هذا الفرز، يقوم أوراكل بتخصيص عدة مقاطع مؤقتة كمساحة عمل لعملية الفرز. يجب أن يكون لديك ملف جداول مؤقت واحد على الأقل لاحتواء هذه المقاطع. إذا لم يكن لديك، فسيكون على أوراكل أن يستخدم ملف جداول النظام. بعد الفرز، يمكن تحرير هذه المقاطع. من ناحية أخرى، سيؤدي هذا الإجراء إلى تجزئة ملف الجداول بالإضافة إلى قصور في الأداء فيَ حالة حدوثه في ملف جداول النظام.

لتجنب هذه المشكلة، تقوم بإضافة ملف جداول مؤقت بواسطة هذا الأمر:

```
CREATE TABLESPACE temporary
  DATAFILE 'C:\ORANT\DATABASE\TEMP.ORA' SIZE 2M REUSE
    AUTOEXTEND ON NEXT 2M MAXSIZE 10M ;
```

منذ صدور أوراكل ٧,٣ توجد طريقة أخرى لمعالجة مساحة الفرز. يمكن أن يتم تخصيص ملف جداول بصورة ظاهرية ليحتوي على sort space (مقطع فرز) واحد. يحدث ذلك من خلال أمر CREATE/ALTER TABLESPACE الذي يحتوي على عبارة TEMPORARY (يجب ألا يتم الخلط بينها وبين الاسم الحقيقي لملف الجداول كما هو محدد في المثال أعلاه). على سبيل المثال، يمكننا تعديل جملة CREATE TABLESPACE بإضافة عبارة TEMPORARY في نهاية الجملة. سيقوم ذلك بإعلام أوراكل بأن ملف الجداول هذا يجب استخدامه للحفاظ على مقطع واحد تم تكوينه عند الفرز الأول على القرص وتم حذفه عند إيقاف الطبعة. يشترك المستخدمون الذين يتم تخصيص عمليات الفرز الخاصة بهم لملف الجداول هذا في هذا المقطع.

تحذير إذا فشلت في تكوين ملفات جداول مؤقتة، سيلجأ أوراكل إلى ملف جداول النظام لتكوين مقاطع مؤقتة.

تشغيل السكريبت

لديك الآن كل المكونات اللازمة لتكوين قاعدة بيانات، لذا يمكنك أخيرا تشغيل السكريبت (انظر التعليمات البرمجية ٣-٣). في المثال أدناه، قمنا بتوضيح كل المكونات معا. عندما تتعلم أو تقوم باختيار سكريبت، قد تحتاج إلى تجربة قطعة واحدة في كل مرة ستجد أنه من الأكثر تحقيقا للفائدة أن تكون لديك السكريبت بأكملها في ملف سكريبت واحد أثناء تشغيلها.

من الممكن أن تستغرق عملية تشغيل سكريبت بأكملها نصف ساعة أو أكثر بناءا على سرعة النظام ومدى تعقيد قاعدة البيانات. ستلاحظ أن السكريبت يحتوي أيضا على أوامر SPOOL و SPOOL OFF. تضمن هذه الأوامر أن كل رسائل المخرجات يتم توجيهها إلى ملف SPOOL (التجميع) create_test.txt في هذه الحالة. بعد إتمام عملية تكوين قاعدة البيانات، يجب إجراء مسح بعناية لهذا الملف لمعرفة ما إذا كانت هناك أى أخطاء.

تحذير لا تقوم جملة CREATE DATABASE فقط بتكوين ملف جداول SYSTEM، بل تقوم بتكوين ملفات التحكم أيضا. يجب التأكد من أن ملفات التحكم هذه غير موجودة قبل تشغيل السكريبت.

تعليمات برمجية ٣-٢

```
SPOOL CREATE_TEST.TXT
CREATE DATABASE test
  LOGFILE
    GROUP 1 ('C:\ORANT\DATABASE\TESTLOG01A.LOG',
          'D:\ORANT\DATABASE\TESTLOG01B.LOG') SIZE 100K
    GROUP 2 ('C:\ORANT\DATABASE\TESTLOG02A.LOG',
          'D:\ORANT\DATABASE\TESTLOG02B.LOG') SIZE 100K
  MAXDATAFILES 20
  DATAFILE 'C:\ORANT\DATABASE\TESTSYSTEM01.DBF' SIZE 25M
  CHARACTER SET WE8ISO8859P1
  ARCHIVELOG
CREATE TABLESPACE test
  DATAFILE 'C:\ORANT\DATABASE\TESTTEST01.DBF' SIZE 1M REUSE
    AUTOEXTEND ON NEXT 1M MAXSIZE 20M ;
CREATE TABLESPACE rollback01
  DATAFILE 'C:\ORANT\DATABASE\TESTROLLBACK01.DBF' SIZE 500K
REUSE
    AUTOEXTEND ON NEXT 500K MAXSIZE 2M
CREATE TABLESPACE rollback02
  DATAFILE 'C:\ORANT\DATABASE\TESTROLLBACK02.DBF' SIZE 500K
REUSE
    AUTOEXTEND ON NEXT 500K MAXSIZE 2M ;
```

```
CREATE TABLESPACE temporary
  DATAFILE 'C:\ORANT\DATABASE\TESTTEMPORORY01.DBF' SIZE 2M
REUSE
    AUTOEXTEND ON NEXT 2M MAXSIZE 10M ;
CREATE ROLLBACK SEGMENT r1 TABLESPACE rollback01
  STORAGE (INITIAL 50K NEXT 50K MINEXTENTS 2) ;
CREATE ROLLBACK SEGMENT r2 TABLESPACE rollback02
  STORAGE (INITIAL 50K NEXT 50K MINEXTENTS 2) ;
ALTER ROLLBACK SEGMENT r1 ONLINE ;
ALTER ROLLBACK SEGMENT r2 ONLINE ;
Rem Ensure that the following line is added to your parameter file
Rem after database creation:
Rem
Rem rollback_segments = (r1,r2)
@C:\ORANT\RDBMS80\ADMIN\CATALOG.SQL ;
@C:\ORANT\RDBMS80\ADMIN\CATPROC.SQL ;
SPOOL OFF
```

تلميح في حالة فشل السكريبت في أي مكان، يمكنك استعادة تشغيله بسهولة. تأكد من أن قاعدة البيانات متوقفة، ثم قم بحذف كل الملفات التي تم تكوينها وابدأ مرة أخرى.

اختبار قاعدة البيانات

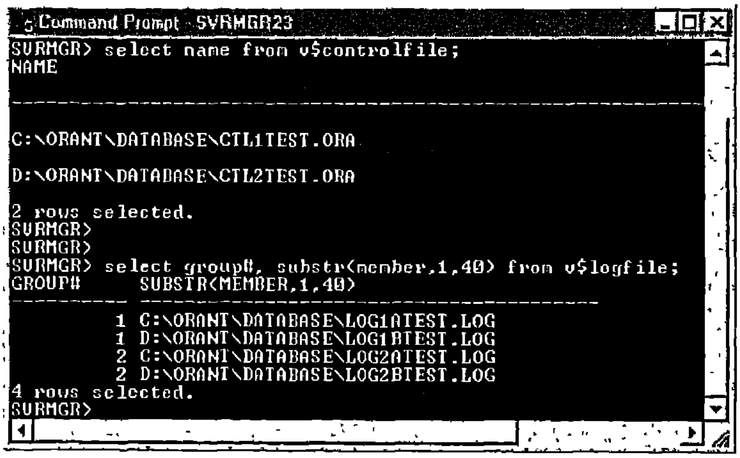
بمجرد إتمام السكريبت، يجب اختبار قاعدة البيانات للتأكد من أن كل شيء يعمل جيدا. إحدى النقاط التي سيتم اختبارها تتضمن اختيار البيانات من الجداول. فيما يلي سرد لبعض الأوامر الرئيسية التي تحتاج إلى معرفتها، سنتناولها بمزيد من التفاصيل في الفصل التالي.

```
SELECT * FROM v$logile ;
SELECT name from v$controlfile ;
ARCHIVE LOG LIST ;
```

يجب أن يعطيك الأمران الأولان نتيجة مشابهة للموضحة في الشكل ٣-٩. ويعطيك الأمر الثاني حالة الحفظ، التي سنناقشها في فصل ٤، إدارة قاعدة البيانات. هناك اختبار آخر هو الاستعلام عن عروض قاموس البيانات dba_tablespaces وdba_rollback_segs.

الشكل ٣-٩

دورة مدير خادم تعرض ملفات الأرشيف والتحكم.

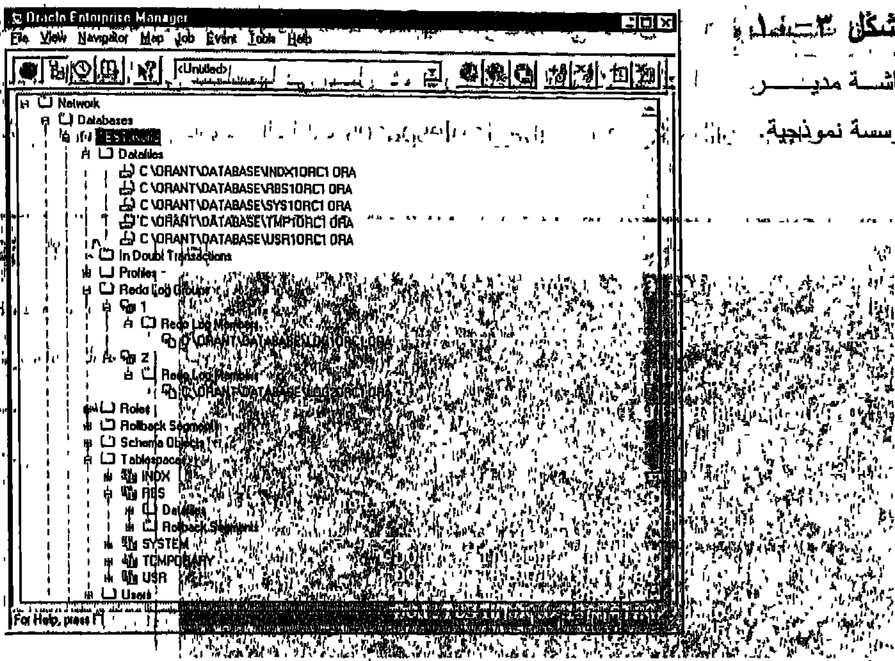


تحذير

إذا كانت المخرجات تبدو معقولة، يجب الآن أن تقوم بإيقاف قاعدة البيانات وعمل نسخة احتياطية لكل ملفات البيانات. من الأهمية بمكان أن تبدأ بنسخة احتياطية جيدة.

إذا كنت قد قمت بإعداد Oracle Enterprise Manager (مدير المؤسسة لأوراكل)، يمكنك استخدامه الآن. يوضح الشكل ٣-١٠ فرز المعلومات التي يمكنك الحصول عليها عن قاعدة البيانات دون أي صعوبات أو سكريبت.

الشكل ٣-١١: شاشة مدير مؤسسة نموذجية.



المستخدمون الافتراضيون

والآن بعد أن قمت بتكوين واختبار قاعدة البيانات، ستكون لديك قاعدة بيانات ذات حسابات عديدة للمستخدمين مثبتة بالفعل بصورة آلية. بدون هذه الحسابات، لن تستطيع استخدامها! هؤلاء المستخدمون الافتراضيون وكلمات السر المقترنة هم:

SYS: الحساب الخاص بمسئول قاعدة البيانات، وتقوم باستخدامه لبدء تشغيل قاعدة البيانات وإيقافها. هو أيضا صاحب قاموس البيانات؛ لذا يجب ألا تستخدمه إلا إذا اضطررت لذلك لأنك قد تتلف أو تدمر قاعدة البيانات. كلمة السر الأولية هي CHANGE_ON_INSTALL.

SYSTEM: الحساب الذي يجب استخدامه لأنشطة DBA العامة. ستقوم معظم عمليات التثبيت بتكوين حساب واحد آخر له امتيازات DBA لمستخدمين محددين. كلمة السر الأولية هي MANAGER.

SCOTT: اسم مستخدم عرض يتم إعداده غالبا مع تثبيت أوراكل. كلمة السر الأولية هي TIGER.

INTERNAL: مرادف لـ sys as SYSDBA، ولا يكون له دائما كلمة سر مقترنة به، ولكن بالتثبيت المخصص الذي قمنا به، تأكدنا من وجود كلمة سر.

لا تستخدم أي حساب من حسابات المستخدمين الافتراضيين أعلاه لتطوير فيجوال بيسك أو قاعدة البيانات. يجب أن يقتصر استخدامها على إدارة قاعدة البيانات فقط. بالنسبة لعملية التطوير، تحتاج إلى إعداد بعض الحسابات الجديدة للمستخدمين، وهذا ما سنتناوله في الفصل ٤.

تحذير: انتبه إلى أن كل قاعدة بيانات أوراكل حديثة التكوين تحتوي على المستخدمين SYS وSYSTEM مع كلمات السر الافتراضية الخاصة بهم كما تم تحديده سابقا. قم بتغيير كلمة السر فورا.

ملاحظة: يعتبر التشغيل والإغلاق ممكنين فقط للمستخدمين الذين لهم دور SYSDBA أو SYSOPER، يتم منح هذه الأدوار من خلال أمر GRANT SYSDBA TO user. يتطلب هذا الأمر وجود ملف كلمة سر خارجي، لكي يقوم المستخدم (بما فيه SYS) بالتسجيل بالدخول بهذا البدور، يجب أن يكون أمر CONNECT هو CONNECT SYS/password AS SYSDBA، حيث أن password هي المجموعة الواحدة في ملف كلمة السر الخارجي. يعتبر هذا الاختلاف البسيط في كلمات السر تمييزا يصعب إدراكه.

تكوين الشبكة

لقد قمنا الآن بتكوين قاعدة بيانات نشطة ويمكنك تشغيلها وإغلاقها. لبدء عمل أي تطوير، ستحتاج إلى أن تقوم بمهمتين إضافيتين. ستحتاج إلى إعداد بعض حسابات مستخدمين للقيام بالتطوير، وستحتاج إلى تكوين الشبكة للسماح للمستخدمين بالاتصال بقاعدة البيانات من أجهزة عن بعد. سنتناول إعداد مستخدمين في الفصل التالي، ولكننا سنناقش أولا موضوعات الشبكة.

يعمل مدير الخادم Server Manager باتصال مباشر بقاعدة البيانات. يعتبر ذلك جيدا طالما أنك على نفس جهاز قاعدة البيانات، ولكن إذا كنت تحتاج إلى الاتصال من جهاز عن بعد، فإنك تحتاج إلى استخدام إمكانيات الشبكة الخاصة بأوراكل. في الواقع، يجب أن تقوم ببعض العمليات

المحلية مثل SQL*Plus ومدير المؤسسة Enterprise Manager باستخدام بعض خصائص الشبكة.

يستخدم أوراكل منهجا متعدد الطبقات في تكوين الشبكة. الطبقة الرئيسية التي سنتعرض لها تسمى SQL*Net لأواكل ٧ أو Net8 لأوراكل٨. عند تثبيت خادم أوراكل، يتم عادة تثبيت هذا البرنامج أيضا. عندما نتناول برامج المزود في الفصل ١١ "تقنية جانب المزود"، سنناقش بالتفصيل البرنامج المكون من طبقات. على أي حال، يجب أن تقوم الآن بإعداد الخادم مع إمكانية معالجة طلبات الوصلات من أجهزة عن بعد.

يستخدم أوراكل عملية المتصنت لانتظار طلبات الوصلات الواردة وتسليمها لقاعدة البيانات المناسبة. يعتبر هذا المفهوم مثل متصنت HTTP في خادم ويب تماما. تحتاج عملية المتصنت إلى ملف تكوين يسمى listener.ora. تحتاج وصلة المزود بقاعدة البيانات إلى ملف تكوين يسمى tnsnames.ora.

ملاحظة أدى إصدار أوراكل ٨ وبرنامج الشبكة Net8 إلى بعض التردد حول نوع البرنامج الذي يجب تشغيله. ببساطة، يستطيع مزود SQL*Net الاتصال بقواعد أوراكل ٧ وأوراكل ٨، لكن عند تثبيت برنامج خادم، يجب استخدام SQL*Net لأوراكل٧ وnet8 لأوراكل ٨. يتوافق متصنتو Net8 مع SQL*Net ٢,٣ بحيث يمكنك استخدامهم للاتصال بقواعد بيانات أوراكل ٧ وأوراكل ٨.

Tnsnames.ora

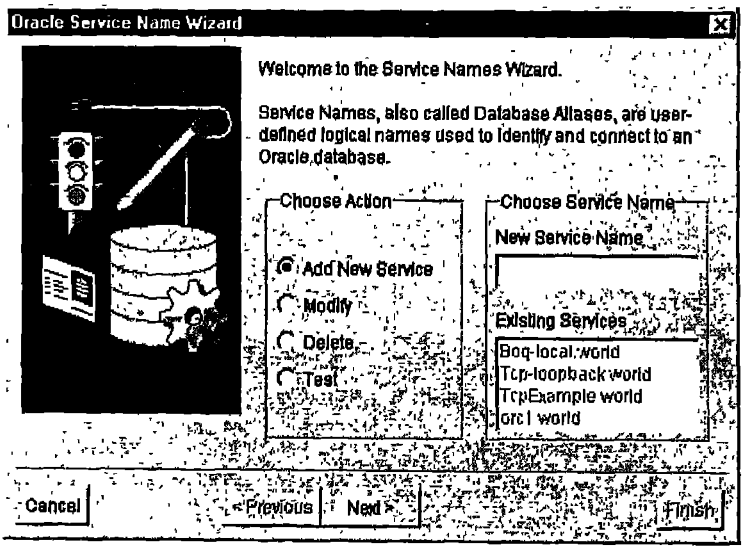
أول ملف تكوين يجب أن نتناوله هو Tnsnames.ora. يستخدم SQL*Net أو مزود Net8 هذا الملف لتحديد قاعدة البيانات التي تريد الاتصال بها. يشبه ملف التكوين هذا ملف hosts، الذي يتم استخدامه لتحليل أسماء المضيفين لعنوان IP. في الفصل ١١ "تقنية جانب المزود"، سنتأكد من أنه يتم تثبيت هذا الملف في قاعدة بيانات المزود. يوجد الملف أيضا على الخادم من أجل بعض طلبات الاتصال الداخلي، هذا هو المكان الذي يتم تكوينه فيه بصورة طبيعية.

هناك بعض الطرق لتوليد ملف التكوين هذا، ولكننا نوصي باختيار واحدة والالتزام بها. سنقوم هنا بمراجعة أداة Net8 Easy Config باختصار، والتي توجد في ويندوز NT. بالنسبة للإصدار ٧,٣ وما قبله، يمكنك استخدام أداة SQL*Net Easy Config.

تلميح حتى إذا لم يكن خـادم أوراكـل قائمـا علـى NT، يجـب تشـغيل NT للإدارة لأن أغلب الأدوات الجديدة يتم تطويرهـا لـهذا النظـام أولا.

عند تشغيل برنامج Net8 Easy Config، سترى قائمة مثل التي توجد في ٣-١١. يمكنك أن تختار تكوين إدخالات جديدة في ملف التكوين، وحذف وتعديل إدخالات قديمة أو مجرد عـرض الإدخالات الموجودة. يسمى كل إدخال database alias (بديل قاعدة البيانات) أي أنـه مؤشـر لقاعدة البيانات التي يمكن استخدامها من قبل مستخدمين عن بعد. ويسمى بديلا لأنه لا يشترط أن يكون sid أي معرف نظام قاعدة البيانات.

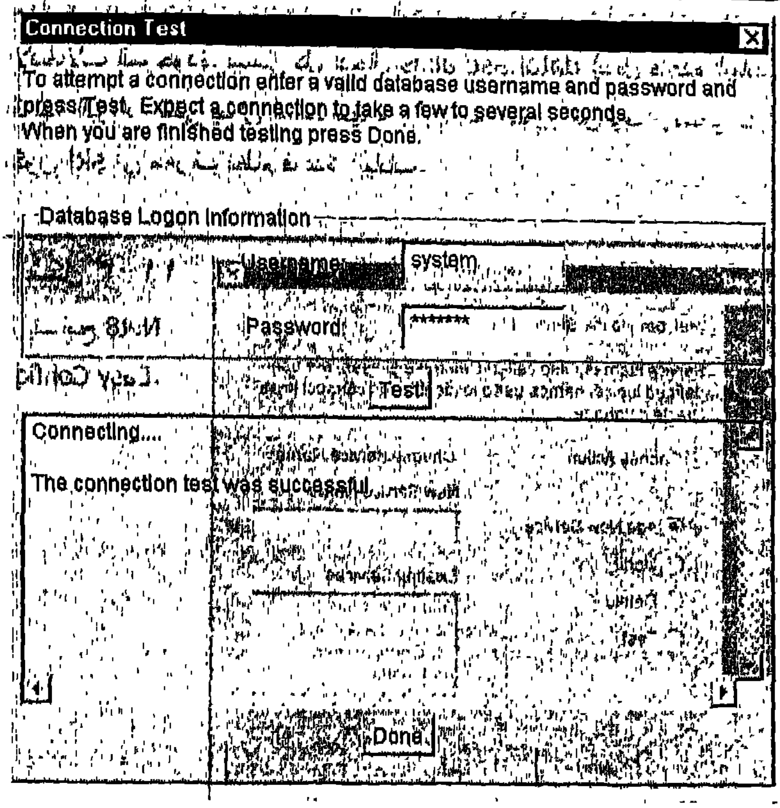
الشكل ٣-١١ برنـامج Net8 Easy Config.



في حالة عرض بدائل قاعدة البيانات، سترى بديلا لقاعدة البيانات ORCL إذا كنت قد قمـت بتثبيت نموذج قاعدة البيانات. يجب أن يقوم كل بديل بتعريف بروتوكول اتصالات الشبكة الـذي يصف البروتوكول اللازم للوصول إليها. على سبيل المثال، يكون أغلب عملي علـى TCP/IP، لذا يذكر هذا البروتوكول في ملف التكوين.

يتم استخدام البروتوكول ذو الاسم bequeath protocol داخليا، على سبيل المثال، عندمـا تكون عملية المستخدم على الجهاز كخادم. يمكن اعتبار بديلا على خـادم ORCL. سـتحتاج أي قاعدة بيانات تريد الوصول إليها من الخادم بواسطة أدوات مثل SQL*Plus إلـى إدخـال بـهذا البروتوكول.

مع Net8 Easy Config، هناك أيضا نافذة في المعالج الذي يسمح لك باختيار الاتصال (انظر الشكل ٣-١٢). تعتبر هذه طريقة جيدة لكي تضمن أنك قد قمت بالإعداد الصحيح لبدائل قاعدة البيانات، بالرغم من أنك تستطيع أيضا تحقيق ذلك بسهولة بواسطة SQL*Plus أو أداة أخرى.



الشكل ٣-١٢: اختبار الوصلة.

يجب أيضا تكوين إدخال لأي قواعد بيانات جديدة تقوم بتكوينها ولبروتوكول الشبكة الذي تقوم بتشغيله. بعد الإنتهاء من ذلك، يجب أن تطلع على ملف التكوين لكي تعتاد عليه. يتم توضيح مثال في التعليمات البرمجية رقم ٣-٤. ستلاحظ، أسفل التعليمات البرمجية إدخالا لقاعدة بيانات الاختبار التي قمنا بتكوينها في الجزء الأول من الفصل. اسم المضيف هو TCCNT01 ومعرف النظام الخاص هو "Test". يقوم أيضا بتعريف منفذي TCP/IP (١٥٢١ و١٥٢٦) التي يتوقع قيام عمليات التنصت بالاستماع عندهما. سيكون ملف tnsnames.ora مختلفا ويعتمد على أسماء قاعدة البيانات وبروتوكولات الشبكة ونسخة خادم قاعدة البيانات.

تعليمات برمجية ٢-٢

```
# TNSNAMES.ORA Configuration file:
C:\ORANT\NET80\ADMIN\tnsnames.ora
# Generated by Oracle Net8 Assistant
Beq-local.world =
  (DESCRIPTION =
    (ADDRESS =
         (PROTOCOL = BEQ)
         (PROGRAM = oracle80)
         (ARGV0 = oracle80ORC1)
         (ARGS =
'(DESCRIPTION=(LOCAL=YES)(ADDRESS=(PROTOCOL=BEQ)))')
    )
    (CONNECT_DATA = (SID = ORC1)
    )
  )
Tcp-loopback.world =
  (DESCRIPTION =
    (ADDRESS =
         (PROTOCOL = TCP)
         (HOST = 127.0.0.1)
         (PORT = 1521)
    (CONNECT_DATA = (SID = ORCL)


TcpExample.world =
  (DESCRIPTION =
    (ADDRESS =
         (PROTOCOL = TCP)
         (HOST = Production1)
         (PORT = 1521)
    )
```

```
    (CONNECT_DATA = (SID = SID1)
    )
  )
orc1.world =
  (DESCRIPTION =
    (ADDRESS =
         (PROTOCOL = TCP)
         (HOST = sccnt01)
         (PORT = 1521)
    )
    (CONNECT_DATA = (SID = ORC1)
    )
```

تحذير بالرغم من أنك تستطيع تحرير هذا الملف يدويا عندما تتعرف عليه، إلا إنه يجب تجنب ذلك بقدر الإمكان. بمجرد إجراء التغييرات اليدوية، لن تستطيع استخدام Net8Easy Config مرة أخرى.

Listener.ora

هناك ملف تكوين آخر أكثر أهمية من ملف tnsnames.ora وهو listener.ora. هذا الملف هو ملف البدء الذي يقوم بتكوين المتنصت، ولا يمكن أن يتم تشغيل هذه العملية بدونه. يحدد الجزء الأول من الملف (انظر الملف المثال في التعليمات البرمجية رقم ٣-٥) عناوين التنصت، أي الخادمين والبروتوكولات وأرقام المنافذ التي ستقوم عمليات التنصت بالاستماع عندها. يتضمن هذا الملف أيضا عنوانين داخليين لا يجب تغييرهما.

يقوم النصف السفلي من الملف بتعريف المعاملات للمتنصت. يوضح المثال أدناه حدود انتهاء الوقت وكلمة سر. يتم استخدام كلمة السر هذه لإثبات شرعية الوصول خلال برنامج lsnrctl. يمكنك أيضا تحديد إمكانية التجميع أو الاستدعاء المسبق لعمليات الخادم هنا. عموما، الجزء الأكثر أهمية هو قائمة SID_LIST_LISTENER التي تحدد معرفي نظام قاعدة البيانات (sids) الذين يتم التصنت عليهم. لإضافة بدائل قاعدة بيانات جديدة، يمكنك استخدام Notepad والتحرير فيه. تأكد من انك تستخدم الصيغة بالضبط في الملف النموذج وفي المثال التالي. أيضا، يجب ملء أي مسافة بيضاء بأحرف مسافات (ASCII بقيمة ٣٢) فقط. مفاتيح الجدولة (ASCII بقيمة ٩) غير مسموح بها)، فهي تمنع بدء تشغيل عملية التنصت.

عندما يقوم مزود مثل فيجوال بيسك بتحديد سلسلة الوصول، يقوم SQL*Net أو Net8 بتمرير هذا التسلسل إلى المتنصت حيث يتم مقارنتها بمعرفي النظام(Sids) الموجودين بالقائمة. في حالة عدم العثور على مثيل، سيتم الإشارة بوجود خطأ، أما في حالة وجود مثيل، سيتم تكوين عملية خادم للتحكم في الوصلة بين فيجوال بيسك وخادم أوراكل نفسه.

يمكن أن يختلف التنفيذ الفعلي لملفي التكوين بناءا على نظام التشغيل. يجب الرجوع إلى التوثيق الخاص بنظام التشغيل الخاص بك. إذا لم يكن لديك خادم NT لأوراكل.

تعليمات برمجية ٣-٥

```
################
# Filename......: listener.ora
# Node..........: local.world
################
LISTENER =
  (ADDRESS_LIST =
      (ADDRESS=
        (PROTOCOL= IPC)
        (KEY= oracle.world)
      )
      (ADDRESS=
        (PROTOCOL= IPC)
        (KEY= ORCL)
      )
      (ADDRESS=
        (COMMUNITY= TCP.world)
        (PROTOCOL= TCP)
        (Host= tssnt01)
        (Port= 1521)
      )
      (ADDRESS=
        (COMMUNITY= TCP.world)
        (PROTOCOL= TCP)
        (Host= tssnt01)
```

```
      (Port= 1526)
      )
   )
STARTUP_WAIT_TIME_LISTENER = 0
CONNECT_TIMEOUT_LISTENER = 10
TRACE_LEVEL_LISTENER = ADMIN
SID_LIST_LISTENER =
  (SID_LIST =
    (SID_DESC =
      (SID_NAME = ORCL)
    )
    (SID_DESC =
      (SID_NAME = ORC1)
    )
  )
PASSWORDS_LISTENER = (oracle)
```

تشغيل عملية التنصت

من الممكن أن تكون العملية التي تتم بين تطبيق مستخدم وقاعدة بيانات أوراكل مربكة قليلا بالنسبة للمبتدئين، ولكن إعتيادها أمر هام. يعتبر ذلك صحيحا على وجه الخصوص عندما لا تسير الأمور كما ينبغي، على سبيل المثال، عندما لا تستطيع الاتصال. في هذه الحالات، يجب تتبع كل الروابط الموجودة في السلسلة ليتم الكشف عن ما لا يعمل منها.

يجب أن يساعد الرسم الموجود في الشكل (٣-١٣) في شرح عملية التصنت. فيما يلي سرد الطريقة عمل هذه العملية:

١- يقوم مسئول قاعدة البيانات (DBA) ببدء برنامج Listener Administration "إدارة التنصت" عن طريق كتابة LSNRCTL عندما يتم بدء قاعدة البيانات.

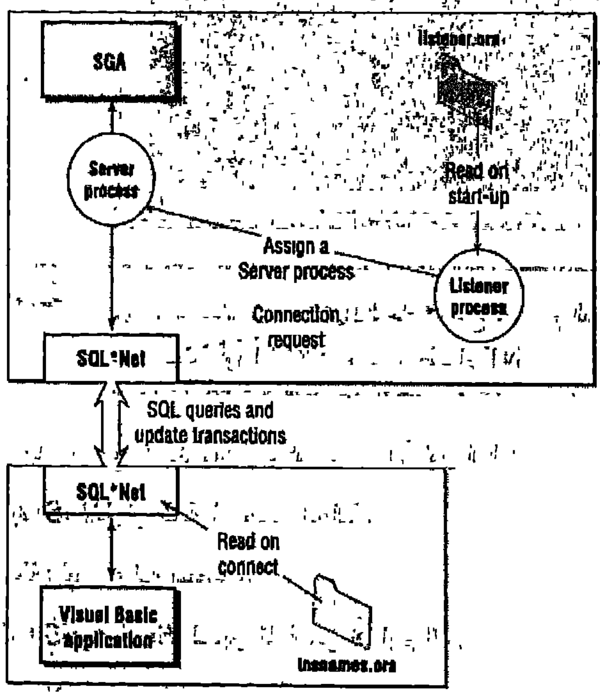
٢- يقوم (DBA) ببدء تشغيل المستمع بواسطة أمر START يقوم هذا الأمر باستدعاء عملية تنصت يوجد تكوينها داخل ملف التكوين listener.ora.

٣- يطلب تطبيق المستخدم (مثل برنامج فيجوال بيسك) وصلة بقاعدة البيانات على Net8 باستدعاء المتنصت.

٤- يقبل المتنصت طلب الوصلة، ثم يقوم بتكوين عملية خادم لمعالجة الطلب.

٥- من خلال عملية الخادم، تتم عملية الاتصال، كما يمكن لتطبيق فيجوال بيسك تنفيذ استعلامات أو حركات تحديث. لا تستخدم دورة المستخدم هذه المتنصت مرة أخرى. يتم عندئذ تحرير المتنصت وينتظر حتى طلب الوصلة التالية.

٦- عندما يقوم تطبيق فيجوال بيسك بإغلاق وصلة قاعدة البيانات، يتم إنهاء عملية الخادم (أو عودتها إلى المجمع).

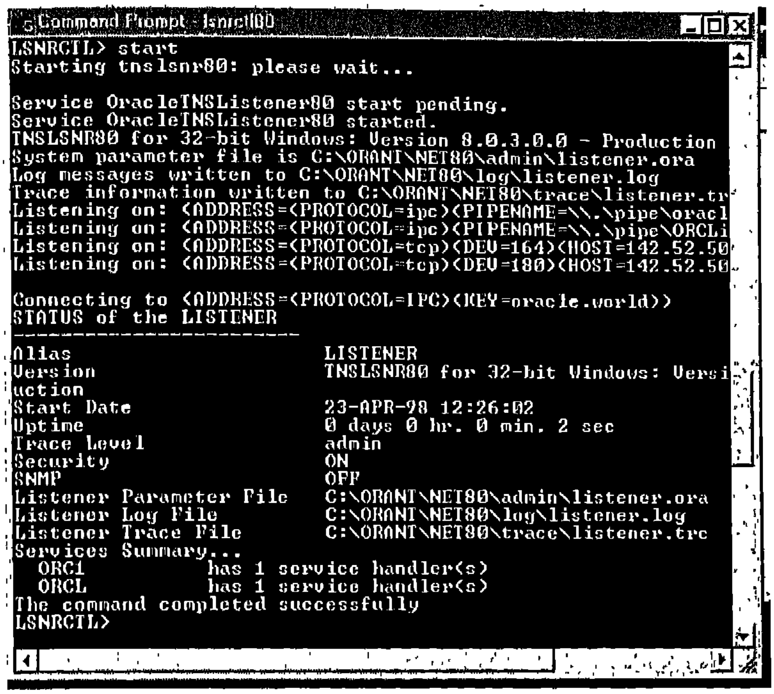


الشكل (٣-١٤) تشغيل عملية التنصت

مراقبة عملية التنصت

بمجرد قيامك بتكوين المتنصت، تعتبر جاهزا لبدئه (راجع الشكل ٣-١٤). لتشغيل برنامج Listener Control (مراقبة المتنصت)، تقوم بكتابة الأمر التالي:

```
C:\>lsnrctl80
```



```
Command Prompt - lsnrctl80
LSNRCTL> start
Starting tnslsnr80: please wait...

Service OracleTNSListener80 start pending.
Service OracleTNSListener80 started.
TNSLSNR80 for 32-bit Windows: Version 8.0.3.0.0 - Production
System parameter file is C:\ORANT\NET80\admin\listener.ora
Log messages written to C:\ORANT\NET80\log\listener.log
Trace information written to C:\ORANT\NET80\trace\listener.tr
Listening on: (ADDRESS=(PROTOCOL=ipc)(PIPENAME=\\.\pipe\oracl
Listening on: (ADDRESS=(PROTOCOL=ipc)(PIPENAME=\\.\pipe\ORCLi
Listening on: (ADDRESS=(PROTOCOL=tcp)(DEV=164)(HOST=142.52.50
Listening on: (ADDRESS=(PROTOCOL=tcp)(DEV=180)(HOST=142.52.50

Connecting to (ADDRESS=(PROTOCOL=IPC)(KEY=oracle.world))
STATUS of the LISTENER
------------------------
Alias                     LISTENER
Version                   TNSLSNR80 for 32-bit Windows: Versi
uction
Start Date                23-APR-98 12:26:02
Uptime                    0 days 0 hr. 0 min. 2 sec
Trace Level               admin
Security                  ON
SNMP                      OFF
Listener Parameter File   C:\ORANT\NET80\admin\listener.ora
Listener Log File         C:\ORANT\NET80\log\listener.log
Listener Trace File       C:\ORANT\NET80\trace\listener.trc
Services Summary...
  ORC1          has 1 service handler(s)
  ORCL          has 1 service handler(s)
The command completed successfully
LSNRCTL>
```

الشكل ٣-١٤
بدء تشغيل عملية التنصت.

ملاحظة تقوم باستخدام أمر lsnrctl7 على جميع نظم التشغيل الأخرى حتى عند تشغيل أوراكل ٧× تحت نظام NT.

عند تشغيل برنامج LSNRCTL، يمكنك كتابة الأوامر التالية:

STATUS: لإظهار حالة المتنصت الحالية.

START: لبدء عملية التنصت.

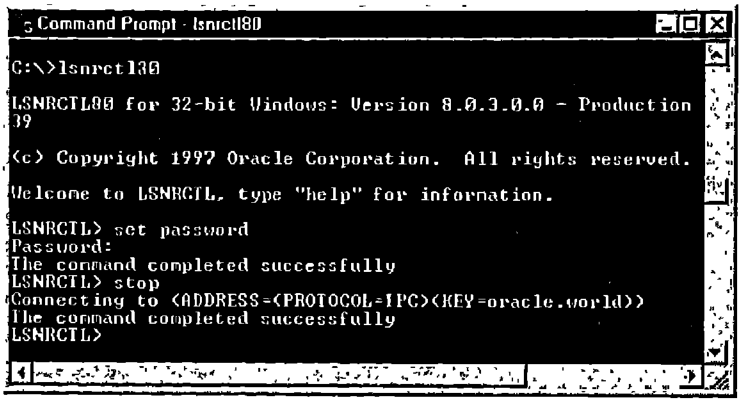
SET PASSWORD: لتعيين كلمة سر إذا لزم الأمر.

STOP: لإيقاف عملية التنصت.

HELP: لإظهار الأوامر الممكنة.

EXIT: للخروج من برنامج Listener Control.

لإيقاف المتنصت. تقوم بكتابة أمر STOP كما هو موضح في الشكل ٣-١٥. إذا كان المتنصت محميا بكلمة سر، يجب استخدام أمر SET PASSWORD يتم تعريف كلمة السر كمعامل تكوين في ملف listener.ora.



الشكل ٣-١٥ إيقاف عملية التنصت.

خلاصة

كانت هذه مقدمة موجزة عن تكوين قاعدة بيانات أوراكل. تم الاطلاع على الأدوات الإدارية المختلفة التي يوفرها لك أوراكل. تم إعداد ملف معامل البدء وإعداد طبعة معطلة عن العمل ثم تكوين ملف سكريبت لتكوين قاعدة البيانات. بالإضافة إلى ذلك، تم التعرف على المستخدمين الافتراضيين لقاعدة البيانات وإعداد المتنصت للسماح لنا بالاتصال من وحدة فرعية عن بعد.

لا تعتبر هذه العملية بسيطة (يعتبر أوراكل عملية تكوين قاعدة بيانات وظيفة متقدمة)، ولكنها عملية يمكنك تنفيذها باتباع التعليمات الموضحة في هذا الفصل. إذا أردت معرفة ما بعد التثبيت، خاصة عندما تريد ضبط التثبيت مع متطلباتك الخاصة، يجب سؤال DBA ذو خبرة أو الحصول على مزيد من الدورات التدريبية بنفسك.

سننتقل إلى تقديم عن إدارة قاعدة البيانات بحيث يمكنك التأكد من أنها آمنة يمكنك الاعتماد عليها. سيضع ذلك الأساس الذي يمكننا بناء التطبيقات عليه.

٤

إدارة قاعدة البيانات

الآن تم إعداد قاعدة بيانات جديدة أو من المحتمل أنك استخدمت قاعدة موجودة بالفعل في موقعك أو قاعدة بيانات العرض التجريبي المرفقة مع أوراكل، يجب إعطاء بعض الوقت للتأكد من صيانتها بصورة صحيحة. لا تعتبر مهام الصيانة من المهام المحببة إلى نفس المطور، ولكن إذا تم تجاهلها، سيتم في النهاية فقدان قاعدة البيانات.

على الرغم من كون أوراكل قاعدة بيانات ثابتة (من الممكن تشغيل النظام لشهور أو سنين بدون حدوث أية مشاكل)، يوما ما سيحدث هناك تلف في القرص، تخطط لعمل نسخ احتياطية وإجراءات الاسترداد. على الرغم من هذا الثبات، يقدر أوراكل عملية الاهتمام بالصيانة الدورية.

أحد المسائل المهمة الأخرى هي السرية. حيث أنه لن يكون هناك أية فائدة من بناء مؤسسة مالية مليئة بأحدث وسائل التكنولوجيا للعملاء إذا نسيت وضع أقفال على صمام الأمن. حيث سيجد شخص ما طريقاً لفعل ما يشاء بالنظام سواء عن عمد أو غير عمد.

لذا هناك حاجة إلى مناقشة المهام التي يتم تنفيذها بصورة منتظمة بواسطة مسئول النظام:

- التأكد من وجود النسخ الاحتياطية في موضعها
- التأقلم مع عمليات الاستعادة
- إعداد سرية قاعدة البيانات
- اتباع نظام إجراءات الصيانة المنتظمة
- استخدام مدير المؤسسة لأوراكل لتبسيط مهام الصيانة

ملاحظة: إذا كانت مسئوليتك جاري تشغيلها بصورة فعالة لقواعد بيانات الشركة، يجب حينئذ أن تكون جاهزاً لشيئين. الأول أنها من الممكن أن تكون عبارة عن وظيفة كل الوقت ولن يكون لديك وقت لتنفيذ التطوير وهو الأمر الذي من الممكن أن يكون ثقيلاً على مطور فيجوال بيسك. ثانية، ستحتاج إلى الكثير من التدريب قدر الإمكان. هذا الفصل هو عبارة عن مقدمة لما يمكن أن يصبح وظيفة. هناك الكثير من الكتب المتخصصة في إدارة قاعدة البيانات لأوراكل، ولكن يجب أيضاً دعم هذه البرامج التدريبية.

إجراءات النسخ الاحتياطي

هناك مدى من البيئات التي تستخدم قواعد البيانات. بعض هذه القواعد يعتبر ضرورياً لإنجاح العمل. يتضمن المخطط غير الحرج النموذجي إدارة الموارد البشرية التي تتعقب كل البرامج التدريبية التعليمية التي حضرها الموظفون. إذا كان سيتم شطبها هذه البيانات كلية، سوف تظل

الشركة قائمة. حتى لو انخفض نظام كشف الرواتب لأيام عديدة، سوف يلاحظ هذا الأمر عد قليل من الأشخاص طالما أن هذا الأمر لم يحد عند خفض الشيكات المدفوعة.

على الجانب الآخر من الأنظمة "المهام الحرجة". هذه هي الأنظمة التي لا يمكن أن يشتغل العمل بدونها: تعتبر هذه القواعد هي العمل نفسه. مثال للشركة بهذا النوع من النظام هوا بيع الكتب باستخدام الويب www.amazon.com. إذا كان هناك تلف في قواعد بيانات Amazon، فإن أي شخص يريد كتاب بصورة ملحة سوف يذهب إلى مكان آخر، هو والأعمال الخاصة به.

إذا فشلت أنظمة المهام الحرجة هذه ولم تستطع الشركة استعادتها بسرعة، سيتم حينئذ فقدان عمل جديد. إذا تم فقدان البيانات، سوف تفقد الشركة المال وكذا احترام العملاء الذين لم يتلقوا أية إجاباتهم على طلباتهم. من الممكن أن يشهد أي شخص كان في مطار عند "عطل" نظام الحجز الفوضى والغضب الذي يولده هذا العطل.

لحسن الحظ، أوراكل مليء بطرق تأكيد إجراءات النسخ الاحتياطي والاستعادة وأيضاً إمكانية العمل ٢٤ ساعة في اليوم. مع ذلك لم توجد هذه الإجراءات بصورة افتراضية، ومن الممكن أن يوفر القليل من المعرفة للنسخ الاحتياطية المباشرة

استيراد وتصدير البيانات

أبسط طريقة لعمل نسخ احتياطية من قاعدة بيانات هو القيام بتصدير البيانات. تعتبر هذه الوظيفة هي نفس الوظيفة التي ستشاهدها في مايكروسوفت أكسس: يتم تصدير البيانات من قاعدة بيانات إلى ملف خارجي يمكن تخزينه في مكان آمن. إذا حدث فقدان لملف بيانات واحد أو أكثر نتيجة لحدوث تلف في القرص، حينئذ يمكنك استعادتها بإجراء استيراد لقاعدة البيانات بالكامل، وهي العملية العكسية للاستيراد.

على العكس من برنامج سطح المكتب الذي يحتوي على عمليات استيراد وتصدير للبيانات، يجب تشغيل أحد البرامج المساعدة الكثيرة لأوراكل المرفقة بقاعدة البيانات. مدير البيانات في مدير المؤسسة لأوراكل هو عبارة وسيط سطح مكتب رسومي بالنسبة للبرامج المساعدة للتصدير/الاستيراد. يقو البرنامج المساعد للاستيراد باستخراج البيانات المحددة من قاعدة البيانات وكتابتها على ملف نظام تشغيل. حينئذ يمكن أن يقوم البرنامج المساعد للاستيراد بقراءة ملف تم تصديره واستيراد البيانات التي يحتويها إما داخل نفس قاعدة البيانات لأغراض الاستعادة أو داخل قاعدة بيانات أخرى، كمثال على الاستخدام الثاني للاستيراد هو نقل جدول جديد من قاعدة بيانات التطوير إلى قادة بيانات الإنتاج.

البرنامج المساعد للاستيراد

يقوم البرنامج المساعد للاستيراد بنسخ وحدات قاعدة البيانات المحددة (أو قاعدة البيانات بالكامل) إلى ملف نظام تشغيل. إذا كنت تقوم بنقل الاستيراد إلى حاسب مختلف، يمكنك حتى تعيين بديل لتنسيق حروف ASCII، مثال ذلك، إذا كنت تقوم بالنقل إلى حاسب IBM رئيسي. قم بتشغيل البرنامج المساعد للاستيراد من موجه سطر الأوامر أو تنشيط أداة مدير البيانات. يختلف الأمر الفعلي مع كل نظام تشغيل. مثال ذلك، إذا كنت تستخدم أوراكل في ويندوز NT، فالأوامر هي: C:\EXP73

بالنسبة لأوراكل ٧,٣ وC:\EXP80 لأوراكل ٨. في أنظمة UNIX وOpen-VMS، سوف يكون "exp" على نحو نموذجي.

هناك ثلاثة طرق يمكنك استخدامها لتصدير البيانات:

- جلسة عمل تفاعلية يتم فيها مطالبتك بكل المعاملات
- طريقة سطر الأوامر التي تقوم فيها بتقديم كل المعاملات في السطر
- طريقة ملف المعامل

كل الطرق الثلاث قابلة لإجراء استيراد، ولكننا نفضل المرونة والتوافق باستخدام ملفات المعاملات. مع ذلك تعتبر جلسة العمل التفاعلية جيدة عند عمل نسخة واحدة فقط لجداول قليلة للانتقال من قاعدة بيانات الاختبار إلى الإنتاج. ملف المعامل هو ملف نص يتم تجميعه مع محرر نص ويحتوي على كلمة أساسية واحدة/زوج من القيم في كل سطر. من أهم المعاملات للمطور هي:

الملف: اسم ملف الاستيراد (افتراضيا لـ expdat.dmp).

الجداول: اسم ملف واحد أو أكثر المراد استيراده (فاصلة مميزة وموضوعة داخل أقواس هلالية عند وجود أكثر من واحدة في جدول).

الفهارس: Y أو N للعلامة إذا أردت حفظ الفهارس مع الجداول.

المدخلات: Y أو N للعلامة إذا أردت حفظ المدخلات مع الجداول.

المنح: Y أو N للعلامة إذا أردت تصدير المنح.

ملاحظة تتاح بعض خيارات الاستيراد فقط من خلال طريقة سطر الأوامر/ملف المعامل. مثال ذلك، من الممكن تنشيط استيراد المسار المباشر فقط بواسطة تمرير DIRECT=TRUE من خلال سطر الأوامر/ملف المعامل.

إذا قمت بالدخول كمستخدم، ستكون قادراً على تصدير أي شيء من المخطط الخاص بك. إذا كانت هناك حاجة إلى تصدير قاعدة البيانات بالكامل، ستحتاج إلى تحديد المعامل التالي:

FULL=Y

المستخدم الذي يطلب استخدام هذا المعامل سيطلب أيضاً دور EXP_FULL_DATABASE. (الدور هو طريقة تقديم امتيازات إلى مستخدم، سنناقش هذا الأمر بتفاصيل أكثر لاحقاً في هذا الفصل "الأدوار المحددة مسبقاً"). كما ستحتاج هذا الامتياز عند التصدير من مخطط مستخدم آخر.

تحذير يمكنك أيضاً إدخال اسم المستخدم وكلمة السر في ملف المعامل، لكن يجب أن تتجنب هذا الأمر قدر الإمكان حيث أن هذا أمر خاص بالسرية: في حالة عدم الإدخال سيطالبك النظام بإدخالها. عند تشغيل ملف Batch، حينئذ يجب القيام بالترميز الدائم لكلمة السر في مكان ما. لذا، يجب أن تحاول إعداد حماية الوصول في ملف Batch على نحو مناسب.

التصدير في وضع التطبيق

لتحقيق التصدير باستخدام ملف المعامل، قم بتحرير ملف يسمى export.par وأدخل النص التالي:

```
   FILE=c:\export\movies.dmp
TABLES= (الأفلام وصور الفيديو والمشتركين)
   INDEXES=Y
   GRANTS=Y
   CONSTRAINTS=Y
```

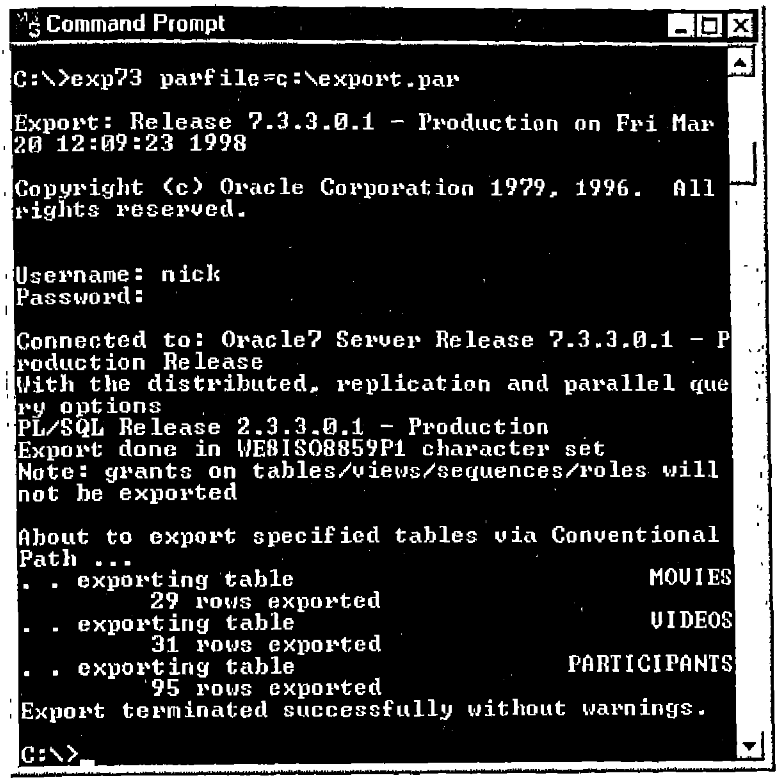
لتنفيذ ملف بنوع

```
   exp73 parfile=c:\export.par
```

المخرجات معروضة في الشكل ٤-١.

الشكل ٤-١

تخصيص امتيازات إجراءات مخزونة للأدوار.



ملاحظة

من الممكن عرض ملف التصدير باستخدام مربع الحوار. إذا قمت بتصدير جدول، يحتوي الملف على نص ASCII لجمل CREATE TABLE وCONSTRAINT مع كل البيانات. كما يمكنك مشاهدة المزايا الممنوحة والأشياء الأخرى الموجودة في الملف. بالإضافة إلى إمكانية نسخ جزء من الملف لإنتاج سكريبت يقوم بإعادة تكوين جدول واحد أو أكثر. مع ذلك، لا تقم بتحرير الملفات وحفظ التغييرات لأنك لن تكون قادراً على استخدامه للاستيراد ثانية.

البرنامج المساعد للاستيراد

من المحتمل انك ستقوم بالاستيراد على أساس منتظم إذا كان هذا جزءاً من إستراتيجية النسخ الاحتياطي، وهو المر الذي يعني أنه نادراً ما سيتم تنفيذ استيراد. مع ذلك، غالباً ما ستستخدم الاستيراد كأنسب طريقة لتحويل البيانات من قاعدة بيانات إلى أخرى، وعندئذ سيتم إجراء استيراد مباشر لملف التصدير.

للاستيراد من تصدير سابق، استخدم البرنامج المساعد للاستيراد. سيعتمد الأمر على الإصدار الجاري تشغيله، في NT سيكون

C:\IMP73

أو

C:\IMP80

كما هو الحال مع التصدير، يمكنك تنفيذ الاستيراد في جلسة عمل تفاعلية أو جملة سطر أوامر أو ملف معامل. من الأفضل وأحيانا من الضروري تشغيل استيراد باستخدام ملف معامل. من اكثر المعاملات استخداما بواسطة المطورين هي File. اسم ملف التصدير (افتراضيا على expdat.dmp).

Tables: اسم جدول واحد أو أكثر المراد استيراده.

Indexes: Y أو N للعلامة إذا أردت استيراد الفهارس مع الجداول.

Grants: Y أو N للعلامة إذا أردت استيراد المنح.

Ignore: Y أو N للعلامة إذا أردت تجاهل رسائل الخطأ نتيجة للوحدات الجاري استيرادها والموجودة بالفعل في قاعدة البيانات.

Commit: Y أو N للعلامة إذا أردت تثبيت التغييرات بصورة أكثر انتظاما.

كما هو الحال بالنسبة للاستيراد، يمكنك أيضا إجراء استيراد كامل للبيانات إذا استخدمت المعامل

FULL=Y

في هذه الحالة، سيتطلب المستخدم دور IMP_FULL_DATABASE لكي يتم منحه بواسطة مسئول قاعدة البيانات DBA.

تلميح يجب الحذر عند استيراد البيانات داخل قاعدة بيانات جديدة، حيث أن الافتراض بالنسبة لكل وحدة قاعدة بيانات أن يتم استيرادها كحركة مفردة. من الممكن أن يضع هذا عبئا ثقيلا على مقاطع إلغاء التعديلات إلى النقطة التي يمكن معها أن يفشل معها الاستيراد. لتجنب هذه المشكلة، يجب استخدام معامل COMMIT=Y. حيث سيضمن هذا تثبيت التغييرات بصورة دورية ولا يتم الاحتفاظ بها كحركة مفردة ضخمة.

الاستيراد في وضع التطبيق

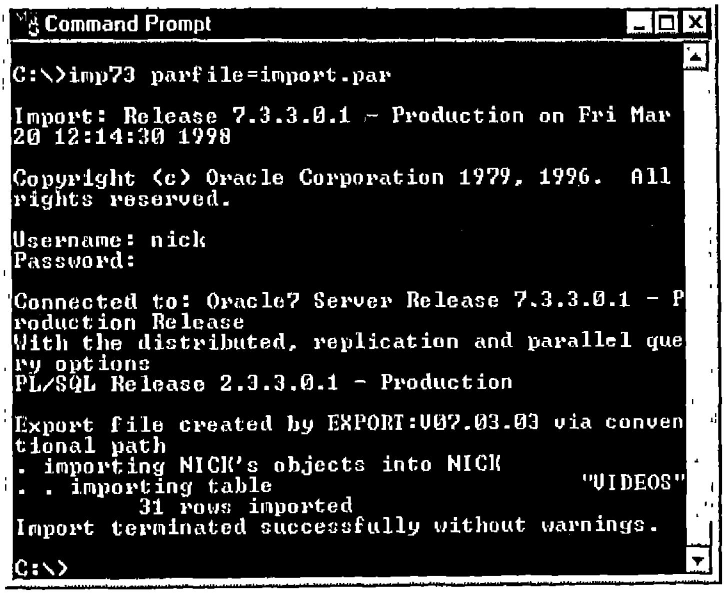
لاستيراد الملف، يجب أولاً إعداد ملف المعامل بنفس الطريقة التي تمت مع الاستيراد. مع ذلـــك، يعتبر هذا من المواقف المفردة، يجب بحث الطور التفاعلي الذي يتم فيه مطالبتك بالمعاملات. قـم بتكوين ملف مثل، import.par كما يلي:

```
FILE=c:\movies.dmp
COMMIT=Y
IGNORE=Y
TABLE=videos
GRANTS=Y
```

ثم يمكنك تشغيله:

```
IMP73 PARFILE=import.par
```

في هذه الحالة، يجري استيراد جدول واحد فقط داخل مخطط مختلف. مثال ذلك في الشـــكل ٤-٢.



```
Command Prompt
C:\>imp73 parfile=import.par

Import: Release 7.3.3.0.1 - Production on Fri Mar
20 12:14:30 1998

Copyright (c) Oracle Corporation 1979, 1996.  All
rights reserved.

Username: nick
Password:

Connected to: Oracle7 Server Release 7.3.3.0.1 - P
roduction Release
With the distributed, replication and parallel que
ry options
PL/SQL Release 2.3.3.0.1 - Production

Export file created by EXPORT:V07.03.03 via conven
tional path
. importing NICK's objects into NICK
. . importing table                       "VIDEOS"
        31 rows imported
Import terminated successfully without warnings.

C:\>
```

الشكل ٤-٢

الناتج مـن دورة الاستيراد.

تلميح إذا كنت تقوم باستيراد كمية كبيرة من البيانات، قم بإسقاط الفهارس داخـــل أي جدول يتم تحديثه. سيؤدي هذا إلى زيادة سرعة الاســـتيراد. يجــب أن يكون لديك SQL سكريبت لإعادة تكوين الفهارس بمجرد انتهاء الاستيراد.

عمليات تصدير النسخ الاحتياطية

يجب الأخذ في الاعتبار أن التصدير ليس هو طريقة النسخ الاحتياطي النموذجي لقاعدة البيانات. لكي تكون محل ثقة على نحو تام، يتطلب التصدير اخذ قاعدة البيانات بصورة منفصلة. لن يكون لدى المستخدمين حينئذ إمكانية الوصول إلى البيانات، وهو الأمر الذي من الممكن أن يكون من الأمور الحرجة. مع ذلك، تتمثل المشكلة الرئيسية في كون النسخة الاحتياطية المصدرة صالحة فقط عند وقت تنفيذ التصدير، أي المرة الأولى التي يقوم فيها المستخدم بتغيير البيانات بعد تشغيل التصدير، ستكون النسخة الاحتياطية قديمة. سبب آخر لعدم كون عمليات التصدير هي استراتيجية النسخ الاحتياطي الرئيسية هو عدم إمكانية تطبيق Redo Logs (سجلات إعادة التشغيل) على التصدير.

تلميح يعتبر تنفيذ التصدير من الطرق النموذجية لتحويل البيانات من قاعدة بيانات إلى أخرى. يمكنك تحديد أية مخططات وحتى أية جداول تريد نقلها.

انعكاس Redo Logs (سجلات إعادة التشغيل)

يعتبر القيام بتصدير دوري أفضل من لا شيء، ولكن من الأفضل استخدامه كخط دفاع ثان في حالة فشل عمليات النسخ الاحتياطي العادي. من أفضل الطرق لإجراء النسخ الاحتياطي لقاعدة البيانات هو استخدام استراتيجية انعكاس ملفات البيانات والحفظ، والذي سنناقشه الآن.

كما رأينا في الفصل ٢ (هيكل قاعدة بيانات أوراكل) يتم تخزين أية تغييرات في قاعدة البيانات كمدخلات سجل إعادة التشغيل Redo Log Entries في فاصل سجل إعادة التشغيل Redo Log Buffer ثم نقلها إلى ملفات سجل إعادة التشغيل المباشر Online Redo Log. تم إعداد أوراكل بصورة افتراضية لمجموعة مفردة من ملفات سجل إعادة التشغيل (ملف عضو واحد لكل مجموعة). في حالة حدوث تلف في القرص الذي يحتوي على هذه الملفات، ربما ستضطر إلى استخدام التصدير لاستعادة قاعدة البيانات، وبالطبع، ستفقد أية بيانات تم تغييرها من تنفيذ التصدير.

إذا أردت تشغيل قاعدة بيانات ثابتة، يجب التأكد من وجود أكثر من ملف عضو لكل مجموعة سجل إعادة التشغيل. بمعنى آخر، يجب انعكاس الملفات. لتنفيذ ذلك، يتم إضافة ملفات أكثر لكل مجموعة، مع ذلك، تحتاج إلى اكتشاف عدد الملفات الموجودة أولا. من الممكن إدخال الجملة التالية في جلسة عمل SQL*Plus إذا كان لديك امتياز مدير قاعدة البيانات.

```
SELECT GROUP# MEMBER FROM v$logfile;
```

```
GROUP#    MEMBER
1         C:\ORANT\DATABASE\LOG1ADEST.LOG
1         D:\ORANT\DATABASE\LOG1BDEST.LOG
2         C:\ORANT\DATABASE\LOG2ADEST.LOG
2         D:\ORANT\DATABASE\LOG2BDEST.LOG
```

يمكنك رؤية وجود مجموعتين فقط من ملفات سجل إعادة التشغيل بعضو واحد في كل منها. تعتبر عملية تقييم العدد الأقصى من المجموعات من أمور الخطأ والصواب. إذا اخترت عدد كبير من المجموعات أو قمت بتكبير مساحة الملفات، حينئذ يتم تضييع مساحة القرص. إذا قمت بتصغير الملفات إلى حد كبير، سيكون هناك عدد كبير من تبديلات السجل بينما يتم انتقال كاتب السجل Log Writer من ملء ملف إلى الذي يليه. في حالة وجود مجموعات قليلة جداً، من الممكن أن تكون هناك فرصة للانتظار.

كل ١٥ إلى ٣٠ دقيقة. يمكنك الحصول على فكرة جيدة حول تسجيل توقيت تبديل السجل من المرات الموجودة في ملفات الحفظ.

إضافة ملفات سجل جديدة

إذا كنت تقوم بإعداد قاعدة بيانات جديدة، تأكد من البدء باستخدام ملفات Mirrored Log. مع ذلك، إذا كنت تستخدم قاعدة بيانات غير معدة بهذه الطريقة، فمن الأمور السهلة نسبيا انعكاس مجموعة موجودة من ملفات Redo Log. في حالة تحديد عدد المجوعات المرادة ومكان وضع ملفات Mirrored Log، حينئذ استخدم الأمر التالي لإعداده في مدير الخادم Server Manager:

```
ALTER DATABASE movies ADD LOGFILE MEMBER
    'd:\redo_logs\log1b.log' TO GROUP 1 ;
```

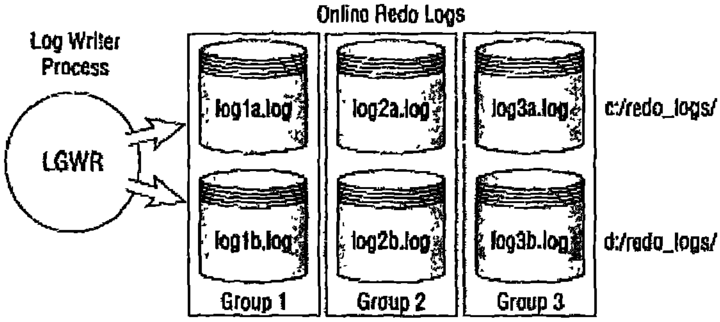
يضيف هذا الأمر عضوا آخر إلى المجموعة رقم ١ الموجودة. نتيجة لوجوب كون ملفات Mirrored Log بنفس الحجم، لا يوجد معامل الحجم في مواصفات الملف. لتكوين مجموعة جديدة، استخدم أمر مختلف بعض الشيء:

```
ALTER DATABASE movies ADD LOGFILE GROUP 3
    ('c:\redo_logs\log3a.log'
    'd:\redo_logs\log3b.log') SIZE 100K ;
```

يجب إعطاء كل ملفات Log نفس الحجم حيث أن الاختلاف بينهم يسبب حدوث ارتباك في الموالفة. يعرض الشكل مخطط الملف الذي يجب تحقيقه عند انعكاس ملفات Redo Log.

الشكل ٤-٣

ملفات سجلات إعادة التشغيل المعكوسة.



تلميح يتم تشغيل ملفات Redo Log بصورة فائقة في قاعدة بيانات نشطة. حين الإمكان، يجب محاولة ترتيب مجموعات الأعضاء على مشغلات قرص مختلفة. مثال ذلك، يجب أن يكون كل الأعضاء "أ" على قرص واحد، وكل الأعضاء "ب" على قرص مختلف. كما يجب فصلهم عن ملفات النظام والبيانات. وهذا يعني أنك بحاجة إلى خمسة مشغلات على الأقل. يتم تزويد معظم أجهزة الخادم، خاصة على NT، باثنين أو ثلاثة من المشغلات، ولكن سيكون الأداء بعيدا عن مستوى الأداء الأمثل.

إذا كان من غير الممكن قيام عملية خلفية LGWR بالكتابة على أي عضو في زمرة Redo Log، ستفشل هذه الطبعة. في حالة تمكن LGWR من الكتابة على عضو واحد في الزمرة على الأقل، حينئذ ستستمر قاعدة البيانات في وضع التشغيل، ولكن سيتم توليد أخطاء في سجل التنبيه مبينة أي من الأعضاء أصابه الفشل. وهذا هو السبب الذي يدعوا إلى فحص سجل التنبيه بصفة يومية.

ملاحظة كما يمكن إضافة ملفات سجلات جديدة باستخدام Backup Manager "مدير النسخ الاحتياطية" في OEM.

انعكاس ملفات المراقبة

ملف المراقبة هو عبارة عن مخزن للكثير من جوانب قاعدة البيانات، بما في ذلك الهياكل الحقيقية (مثل الجداول) والوضع الحالي لملفات Redo Log (سجلات إعادة التشغيل). يتم فتح ملفات المراقبة عند بدء تشغيل طبعة قاعدة البيانات وكانت مستمرة في الفتح والتحديث حتى يتم إغلاق الطبعة. لا يمكنك عمل نسخة من ملف مراقبة باستخدام أمر نظام تشغيل مثل Copy بينما يجوي تشغيل الطبعة، على الرغم من وجود أمر SQL يسمح لك بتنفيذ ذلك:

ALTER DATABASE BACKUP CONTROLFILE TO 'control2.ctl';

يجب استخدام هذا الأمر في الحالات الطارئة فقط، لأنه بمجرد حدوث أية تغييرات في قاعدة البيانات، تصبح النسخة الاحتياطية متقادمة. تعتبر هذه النسخة مفيدة بصورة أساسية في حالة وجود مشكلة رئيسية مع قاعدة البيانات وكنت على وشك على إغلاقها، ولكن تريد التأكد من وجود ملف مراقبة صحيح.

من الممكن حدوث مشاكل عند فقدان ملف المراقبة. وكنتيجة لسهولة الحل، يجب التأكد من وجود ملفي مراقبة نشطين على قرصين مختلفين. ومن باب الحذر قم بعمل ثلاثة نسخ على ثلاثة أقراص مختلفة. لسوء الحظ، الوضع الافتراضي لأوراكل هو ملف مراقبة مفرد.

ملاحظة هناك أعباء إضافية قليلة في عملية انعكاس ملفات المراقبة نتيجة لعدم كونها مكثفة من ناحية الكتابة.

إذا لك تكن قد قمت بإعداد قاعدة بيانات خاصة بك، قم أولاً بفحص كم عدد ملفات المراقبة الموجودة لديك وما هي أسمائها. يمكنك القيام بذلك باستخدام الأمر التالي في Server Manager "مدير الخادم":

```
SELECT * FROM v$controlfile;
c:\data\ORA_CONTROL1.CON
d:\data\ORA_CONTROL2.CON
```

يجب إضافة ملف جديد في حالة وجود ملف واحد فقط:

١- أغلق قاعدة البيانات.

٢- انسخ ملف المراقبة الحالي إلى موقع ثان من الأفضل إلى قرص مختلف.

٣- قم بتحرير ملف init.ora لتحديد ملف مراقبة جديد، مثال ذلك:

٤- control_files=(c:\oradata\test\control1.con,d:\oradata\test\control2.con).

٥- أعد بدأ قاعدة البيانات.

٦- أعد تشغيل أمر SELECT المعروض أعلاه للتأكد من وجود أكثر من ملف مراقبة.

ملاحظة تذكر أنه يتم فتح ملفات المراقبة عند بدأ تشغيل قاعدة البيانات مع الاستمرار في الكتابة عليها في أي مرة يحدث فيها تغيير. لا يمكن أخذ نسخة من ملف المراقبة في حالة تشغيل قاعدة البيانات.

إذا قمت بانعكاس ملفات المراقبة بهذه الطريقة، وأصبح أحدهم تالفاً أو تم حذفــه مصادفــة، يمكنك دائماً استخدام الملف المتبقي كمصدر لنسخة حديثة من ملف المراقبة. عند فقــدان ملــف المراقبة تظل قاعدة البيانات موجودة، ولكن لا يمكن القيام بأية إجراءات أخرى في قاعدة البيانات بمجرد اكتشاف أوراكل فقدان ملف المراقبة. عند إغلاق قاعدة البيانات، لا يمكن إعادة بدئها حتى يتم استعادة ملف المراقبة. في حالة فقدان ملف مراقبة وأردت استعادة قاعدة البيانات ثانية، يجـب العثور على موقع مختلف لملف المراقبة مع تعديل ملف البدء لتعريف الموقع الجديد.

تتسبب عملية فقدان أية ملفات مراقبة في تصرف أوراكل بنفس الطريقـــة: ظـــهور قـــاعدة البيانات، ولكن مع عدم إمكانية إحداث أية تغييرات. في حالة فقدان كل ملفات المراقبـــة، هنـــاك طريقة واحدة لاستعادة ذلك والتي لا تتطلب استعادة كاملة من النسخة الاحتياطية القديمة. تشـــبه الصياغة تلك الموجودة فـــي جملـــة CREATE DATABASE. عنـــد تشـــغيل أمـــر ALTER DATABASE BACKUP CONTROLFILE TO TRACE، يتـــم توليـــد جملـــة CREATE CONTROLFILE بواســـطة أوراكـــل ووضعـــها داخـــل دليـــل ممثـــل فـــي معـــامل البـــدء user_dump_dest. من الممكن استخدام هذه الجملة لإعادة تكوين ملفات المراقبة.

ضمان فعالية عمليات الحفظ

والآن بعد إعداد ملفات سجلات الإعادة المعكوسة، تحتاج إلى التأكد من سريان عمليات الحفـــظ. بدون حفظ ملفات سجلات الإعادة، تعتبر عملية استعادة تلف الوسيط إلى نقطة الفشل من العمليات المستحيلة من الناحية الفعلية. يجب اتباع عملية مكونة من خطوتين هما: تغيير

طور قاعدة البيانات وتعديل ملف البدء. أولاً، تحتاج إلى تغيير قاعدة البيانات بحيث تصبـــح لديها القدرة على الاستعادة من قرص تالف. لتنفيذ ذلك، يجب التأكد من حفظ ملف السجل قبـــل الكتابة فوقه. يحتفظ أوراكل بسجل تتبع لكل ملفات السجل المحفوظة باستخدام رقم مسلســـل. إذا كانت هناك حاجة في أي وقت إلى استعادة قاعدة البيانات، يعرف أوراكل ما إذا كانت أية ملفـات محفوظة مطلوبة بالإضافة إلى ملفات السجل الحالية. لوضع قاعدة البيانات في طور الاســـتعادة، استخدم الأمر التالي في مدير الخادم:

```
ALTER DATABASE ARCHIVELOG;
```

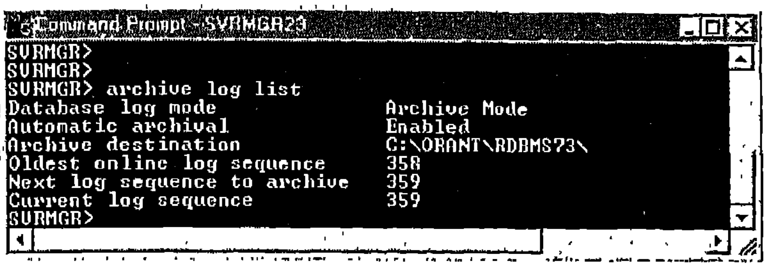
لاستعراض ما إذا كانت قاعدة البيانات في طور الاستعادة يمكنك استخدام الأمر التـــالي مـــن مدير الخادم ولكن عندما تكون قاعدة البيانات في حالة بدء التركيب.

قائمة السجلات المحفوظة

يجب أن تعرض هذه القائمة نتائج شبيهة بالشكل ٤-٤. أعط اهتمام خاص إلى طور سجل قـــاعدة البيانات وحالة الحفظ الآلي.

الشكل ٤-٤

تأكيد موقف الحفظ في قاعدة البيانات.



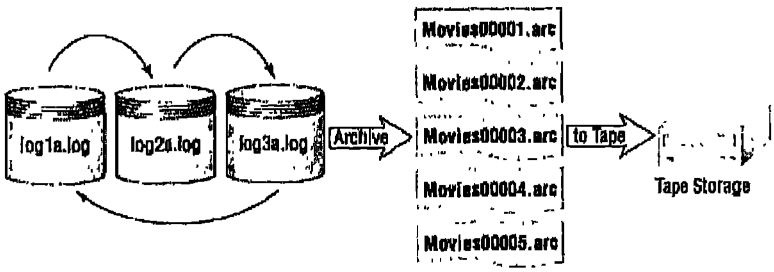
تعتبر عملية إعداد طور الحفظ لقاعدة البيانات هي الخطوة الأولى من العملية. إذا لم نبدأ عمليات الحفظ، لن يتم حفظ أي من ملفات سجلات الإعادة آليا. نتيجة لعدم كتابة أو راكل فوق ملف غير محفوظ في طور الاستعادة لن يتم إيقاف قاعدة البيانات في النهاية. للتأكد من تشغيل الحفظ، تحتاج إلى إضافة سطر واحد أو أكثر إلى ملف البدء كما هو معروض في هذا المثال:

```
log_archive_start = TRUE
log_archive_dest = e:\archives\
log_archive_format = "Movies%S.ARC"
```

السطر الأول هو الأكثر أهمية حيث هو المسئول عن بدء عملية الخلفية للحفظ (ARCH) عند بدء طبعة قاعدة البيانات (راجع الشكل ٤-٥). يحدد السطران الاختياران الآخران مكان وضع الملفات المحفوظة (في هذه الحالة، في e:\archives\directory) مع تخطي خواص تسمية الملف الافتراضية. من الأفضل ترك هذا السطر، ولكن في حالة استخدامه تأكد من عدم تغييره مطلقا. لتنشيط الحفظ، يجب إيقاف قاعدة بيانات سارية وإعادة تشغيلها باستخدام ملف البدء.

الشكل ٤-٥

حفظ ملفات سجلات إعادة التشغيل.



سيتم اشتقاق أسماء ملفات السجل المحفوظة من معامل log_archive_format في ملف البدء. في الحالة المعروضة سابقا يوفر "%S" رقم مسلسل بحيث يمكن الإخبار عن ترتيب الملفات المحفوظة، وباستخدام هذان سترى ملفات مثل الآتي:

Movies00001.ARC,
Movies00002.ARC,
Movies00003.ARC,

كما سترى في مرحلة الاستعادة، يستخدم أوراكل هذه الرقاة المسلسلة للتحديد الدقيق لأية ملفات مطلوبة لاستعادة قاعدة بيانات تالفة.

حين الإمكان، يجب أن تكون السجلات على قرص مختلف عن أي ملف من ملفات سجلات الإعادة، ولكن لا يعتبر هذا من الأهمية بمكان في حالة التأكد من عمل نسخة احتياطية لقاعدة البيانات أيضا، وهي المهمة التي سنغطيها في القسم التالي. كما يجب عمل نسخ احتياطية للسجلات لأداء التسجيل الليلي، في حالة تلف القرص. في بعض البيئات، يمكنك عمل نسخ احتياطية على الشريط مباشرة. تعتبر ملفات سجل الإعادة المحفوظة هذه ضرورية لعملية الاستعادة، لذا يجب بذل كل مجهود للتأكد من جدوى هذه الملفات في وسيط التخزين المجمع.

ملاحظة يجب المحافظة على طول السجلات بدرجة كافية لاستعادة احدث نسخة احتياطية خاصة بك. بالنسبة لي أنا شخصيا فأنا أفضل الاحتفاظ بملفات كافية على قرص لآخر ثلاث نسخ احتياطية للطوارئ. من أفضل الطرق لاختبار عملية الحفظ هو إجبارها على الظهور. عند امتلاء ملء ملف سجل الإعادة، تقوم عملية كاتب السجل بالتبديل إلى ملف سجل الإعادة التالي. يؤدي الأمر التالي إلى إجبار حدوث هذا بطريقة مباشرة.

تبديل ملف سجل تبديل النظام بعد تنفيذ هذا الأمر، افحص وجهة الحفظ: يجب أن تشاهد ملف جديد، إذا لم يحدث ذلك، يجب مراجعة تكوين خاصية الحفظ. فقط.

ما هو عدد مشغلات القرص التي تحتاجها؟

لقد تكررت في مرات عديدة في هذا الكتاب أن أوراكل يفضل أقصى عدد من مشغلات الأقراص يمكنه الحصول عليه، ولكن ما هو عدد هذه المشغلات؟ من الأمور الجديرة النظر إلى النسق الأمثل للأقراص في قاعدة بيانات أوراكل القواعد العامة:

- يجب أن يكون برنامج نظام أوراكل على قرص واحد.
- يجب أن يكون ملف جداول النظام على قرص مختلف.
- يجب انعكاس ملفات سجل الإعادة كما يجب أن تكون كل مجموعة أعضاء على أقراص مختلفة.

- يجب أن تكون مقاطع إلغاء التعديلات على قرصين مختلفين.
- يجب أن تكون المقاطع المؤقتة على قرص منفصل.
- يجب أن تكون ملفات وفهارس البيانات على أقراص مختلفة.

يجب انعكاس ملفات المراقبة، ولكن نتيجة لكونها غير مكثفة من ناحية الإدخال والإخراج، ولكن يمكن وضعها في مكان آخر. ما زال هذا الإجراء يغطي تسعة أقراص بدون محاولة. من الممكن أن نريد المزيد، إذا زادت ملفات البيانات في الحجم وتطلبت المزيد من المجهود.

من الناحية العملية، نادرا ما يتحقق مستوى الفصل، ويجب إدراك أكثر الأقراص استخداما لمعالجة هذا الأمر بصورة صحيحة تحتاج إلى التأقلم مع مراقبة والتوائم مع قاعدة بيانات أوراكل، خاصة قبل القيام باستثمار ضخم في الأجهزة.

تنفيذ النسخ الاحتياطية السريعة

حتى الآن لقد أعددنا قاعدة البيانات الخاصة بنا بحيث يتم انعكاس ملفات سجل الإعادة وملفات المراقبة. يتم إعداد قاعدة البيانات لحفظ ملفات سجل الإعادة. في حالة فشل قرص (ويجب القول متى يتم ذلك)، ستكون الآن جاهزا لاستعادة الملفات التي فقدتها. في الحقيقة، سيستمر أوراكل في التشغيل عدا إعادة تسجيل المشكلة في حالة التسجيل. من الممكن حتى مع المراقبة الآلية ترقيم الصفحات. مع ذلك، لن يتم إجراء أية حركة مقابل ملف البيانات التالف، كما ستحتاج طبعة قاعدة البيانات إلى استعادة الوسائط. لا يحتوي أوراكل على انعكاس لملفات قاعدة البيانات وإذا لم يكن لديك مربع RAID لتوفير انعكاس الأجهزة، تحتاج إلى استراتيجية موضوعة لعمل نسخ احتياطية لهذه الملفات بصفة دورية. إذا كان لديك نسخ احتياطية صحيحة (بما في ذلك ملفات سجل الإعادة)، يمكنك استعادة قاعدة البيانات بصورة دائمة.

من الممكن أن تكون النسخة الاحتياطية لملفات قاعدة البيانات من الأمور البسيطة جدا. بأي طريقة تختارها للتنفيذ، سوف تستخدم أوامر نظام التشغيل (سواء كان هذا الأمر نسخة أو أمر نسخ احتياطي مركب). لنقل ملفات البيانات إلى ملف حفظ أو شريط. تسمح لك بعض البيئات بإجراء نسخة احتياطية من نظام تشغيل حتى لو كانت الملف مفتوحا ويجري الكتابة فيه، ولكن لا يجب استخدام هذه الطريقة مطلقا لأن النتائج ستكون غير متوافقة وغير موثقة حتى لو تمكنت من تشغيلها. أبسط طريقة لإجراء نسخة احتياطية هي إغلاق قاعدة البيانات، عمل نسخ احتياطية لكل الملفات ثم إعادة تشغيل قاعدة البيانات. يسمى هذا بالنسخ الاحتياطي البطيء. من الممكن أن يؤخذ هذا ساعة أو أكثر بناءا على أحجام الملفات، ولذا يتم إجراء هذا النوع في حالة عدم وجود أي نشاط وبخاصة النشاط المفاجئ.

مع ذلك، يعطيك أوراكل يعطيك أوراكل أسلوب أفضل يسمى بالنسخ الاحتيـــاطي الســـاخن. بواسطة إعداد كملف جداول بالمقابلة بطور نسخ احتياطي، يمكنك نسخ عرض متوافق للجـــداول بالنسبة لقرص أو شريط النسخ الاحتياطي بدون إغلاق قاعدة البيانات. لتنفيـــذ ذلـــك، اســتخدم الأوامر التالية:

```
ALTER TABLESPACE SYSTEM BEGIN BACKUP;
```

نسخة النظام التشغيل المضيف أو أمر النسخ الاحتياطي

```
ALTER TABLESPACE SYSTEM END BACKUP;
```

تلميح من الممكن تنفيذ أي أمر نظام تشـــغيل ، مـــن ســـكريبت SQL باســـتخدام حرف"$" قبل الأمر.

يجب أن يكون أمر BACKUP قادرا على تشغيل في ملف مفتوح. يتم عرض مثـــال إنتـــاج سكريبت لـــ OpenVMS أدناه:

```
ALTER TABLESPACE SYSTEM BEGIN BACKUP;
$backup/ignore=(nobackup,interlock)/log -
  oracle_db_0:[TBS]ORA_SYSTEM.DBS; oracle_backup
ALTER TABLESPACE SYSTEM END BACKUP;
ALTER TABLESPACE ROLLBACK01 BEGIN BACKUP;
$backup/ignore=(nobackup,interlock)/log -
  oracle_db_1:[TBS]ROLLBACK01.DBF; oracle_backup
ALTER TABLESPACE ROLLBACK01 END BACKUP;
ALTER TABLESPACE ROLLBACK02 BEGIN BACKUP;
$backup/ignore=(nobackup,interlock)/log -
  oracle_db_2:[TBS]ROLLBACK02.DBF; oracle_backup
ALTER TABLESPACE ROLLBACK02 END BACKUP;
ALTER TABLESPACE TBS_INDEX BEGIN BACKUP;
$backup/ignore=(nobackup,interlock)/log -
  oracle_db_1:[TBS]TBS_INDEX.DBF; oracle_backup
ALTER TABLESPACE TBS_INDEX END BACKUP;
ALTER TABLESPACE TBS BEGIN BACKUP;
$backup/ignore=(nobackup,interlock)/log -
  oracle_db_2:[TBS]TBS.DBF; oracle_backup
```

```
ALTER TABLESPACE TBS END BACKUP;
ALTER DATABASE BACKUP CONTROLFILE TO 'oracle_backup:TBS_1.ctl';
ALTER DATABASE BACKUP CONTROLFILE TO TRACE;
```

تحـذيـر: لا تسمح ويندوز NT بإجراء نسخ احتياطية على ملفات مفتوحة من خلال البرامج المساعدة أو أوامر نظام التشغيل. للتعامل مع ذلك، يجب استخدام البرامج المساعدة OCOPY73.EXE أو OCOPY80.EXE. وذلك بإجراء نسخة ثنائية، مع ذلك فمن الأفضل والآمن القيام بإغلاق قاعدة البيانات وعمل نسخة عادية باستخدام NTBACKUP.

يقوم هذا الأمر بعمل نسخ احتياطية كل نسخة من ملفات الجداول الخمسة على قرص. (يوجد ملف واحد فقط لكل ملف جدول). بمجرد الانتهاء من النسخة، يجب إنهاء طور النسخ الاحتياطي في ملفات الجداول بأسرع وقت ممكن وإلا سيتأثر مستوى الأداء بدون داع. من الممكن أن يكون السطران الأخيران زائدين، ولكنها ستقوم بأداء نسخة احتياطية فورية لملف التهيئة. يتم قراءة هذا الملف عند بدأ التشغيل فقط، ولا يتم تغييره، يفضل بعض مسئولو قاعدة البيانات مضاعفة الحذر وعمل نسخ احتياطية لكل شيء، وإذا قمت أنت شخصيا بذلك فلن يسبب لك ذلك أية مشاكل مطلقا.

قائمة اختبار النسخ الاحتياطي

فيما يلي ملخ النقاط التي ستساعدك في التأكد من إجراء النسخ الاحتياطية بشكل صحيح:

١- تأكد من انعكاس ملف المراقبة.

٢- تأكد من انعكاس ملفات سجل المراقبة.

٣- تأكد من تشغيل وسريان الحفظ.

٤- قم بتنفيذ تبديل سجل مع التأكد من حفظ ملفات سجل الإعادة.

٥- أغلق قاعدة البيانات وقم بعمل نسخة احتياطية كاملة. تعتبر هذه الخطوة من الخطوات الأساسية لأنه لا يمكن استخدام النسخ الاحتياطية السابقة لتبديل سجل حفظ قاعدة البيانات ALTER DATABASE ARCHIVELOG لتنفيذ الاستعادة الكاملة لقاعدة بيانات.

٦- قم بإعداد أمر شغل مجموعة عادي لتنفيذ عمليات النسخ الاحتياطي السريع وتخزين ملفات سجل إعادة الحفظ على شريط.

استعادة قاعدة بيانات

بمجرد وجود كل عمليات النسخ الاحتياطي سارية ومعدة بصورة مناسبة، تعتبر الآن جاهزا لأية كارثة من الممكن أن تحدث. بالطبع في حالة حدوث الكارثة، عادة ما يتم تلف نموذج القرص، وسف يكون هذا الوقت في غير محله على الإطلاق. إذا كانت قاعدة البيانات غير حرجة، سيكون لديك وقت لفحص المشكلة وإحضار مسئول قاعدة بيانات أكثر خبرة.

أي كان عدد قواعد البيانات الضرورية للمهام، يجب الاستمرار في أعمال الشركة سواء كان هذا نظام حجز تذاكر طيران أو نظام طلب فوري على الويب. لاستعادة قاعدة البيانات، يجب أن تكون في موقع الحدث، حيث أنه إذا لم تقم بإعداد استراتيجية النسخ الاحتياطي بشكل صحيح فستضاعف من حجم الجهد المبذول.

أساسيات الاستعادة

لقد ناقشنا في الفصل ٢، بنية قاعدة بيانات أوراكل، العملية التي يتم فيها تسجيل التغييرات في قاعدة البيانات كمدخلات سجل الإعادة في الذاكرة الوسيطة لسجل الإعادة ثم النقل إلى ملفات سجل الإعادة الفورية بواسطة عملية كاتب السجل (LGWR). ثم يتم حفظ هذه الملفات على قرص أو شريط باسطة عملية الحافظ (ARCH). تزود توليفة النسخ الاحتياطية لملفات البيانات والنسخ الاحتياطية لملف سجل الإعادة أوراكل بما يحتاجه لكي يتمكن من استعادة قاعدة بيانات إلى ما قبل نقطة الفشل بالضبط.

في حالة فقدان قرص بملف بيانات ينتمي إلى ملف جداول SYSTEM به، يجب إغلاق قاعدة البيانات يدويا باستخدام أمر SHUTDOWN ABORT. من الممكن إحضار ملفات جداول Non-SYSTEM بصورة منفصلة (واستعادتها) بدون التأثير على موقف ملفات الجداول الأخرى. في حالة فقدان كل أعضاء زمرة ملف سجل أو عملية خلفية إلزامية، ستنتهي طبعة قاعدة البيانات آليا. يجب مراجعة الموقف ومحاولة العثور على سبب للتلف. من أكثر المخططات المحتملة هو قرص تالف ويجب أن يكون هذا واضحا بدرجة كافية بواسطة رسائل النظام وبمجرد النظر على النظام. يجب إصلاح القرص أو استبداله، بحيث يكون من المهم وجود أجهزة إضافية متاحة.

تحذير من المواقف التي واجهتها عدم إمكانية وجود أجزاء إضافية لكل أنواع الأقراص في التركيب. يعني تلف القرص وجوب استعادة ملفات النسخ الاحتياطي على قرص مختلف. تعتبر هذه عملية استعادة أكثر تعقيدا والتي تستوجب إعادة تسمية ملفات الجداول أو تكوين متغير بيئة يشير إليها. من الأفضل تجنب هذا الموقف حتى تتوافر لديك الخبرة الكافية كمسئول قاعدة بيانات حيث يعتبر تنفيذ العملية أثناء استعادة الضغط العالي ليس بالوقت المناسب للتعلم. من الممكن أن تريد وجود مسئول قاعدة بيانات.

الاستعادة

إذا كنا سنأخذ كل مدخلات الإعادة وتطبيقها على نسخة احتياطية سليمة، يمكننا إعادة تكوين قاعدة البيانات إلى النقطة التي تم فيها تلف القاعدة. هذا هو بالضبط ما يقوم به استعادة أوراكل، وأفضل ما به أن معظمه يتم معالجته آليا "راجع الشكل ٤-٦". خطوات استعادة البيانات هي كما يلي:

١- التأكد من إغلاق قاعدة البيانات. (استخدم ABORT إيقاف إذا اضطررت إلى ذلك).

٢- التأكد من إصلاح أي عطل في الأجهزة.

٣- البحث عن آخر نسخة احتياطية لقاعدة البيانات واستعادة أية ملفات بيانات مفقودة.

٤- في حالة فقدان أي ملف مراقبة، قم بنسخ الإصدار المنعكس إلى القرص الذي تم إصلاحه.

٥- تأكد من وجود كل ملفات سجل الإعادة المحفوظة من شريط أو قرص (مع ذلك سيطالبك أوراكل بذلك في حالة عدم التوافر). من الأفضل وجود مساحة على وجهة الحفظ. الكتابة السريعة.

٦- قم ببدء الاستعادة بواسطة كتابة الأوامر التالية في مدير الخادم أثناء تسجيل الدخول على SYS (أو المرادف INTERNAL).

```
SVRMGR> STARTUP MOUNT PFILE=c:\orant\database\inittest.ora
SVRMGR> SET AUTORECOVERY ON
SVRMGR> RECOVER DATABASE ;
```

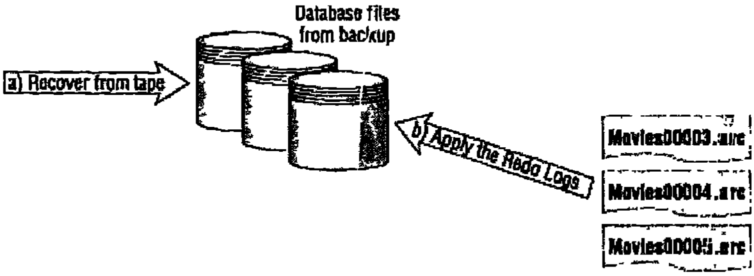
٧-عند إتمام الاستعادة، يمكنك بدء تشغيل قاعدة البيانات باستخدام

```
SVRMGR> ALTER DATABASE OPEN;
```

بعد إتمام كل العمل المتضمن في إعداد قاعدة البيانات لتحقيق انعكاس ملفات المراقبة وسجلات الإعادة وضمان حفظها، يتم تجديد هذه القاعدة لاستعراض إمكانية إنجاز الاستعادة بصورة مبسطة للغاية.

الشكل ٤-٩

استعادة قاعدة بيانات محفوظة من ملفات سجلات إعادة التشغيل المحفوظة



تلميح كلما زادت نسبة عمل النسخ الاحتياطية لقاعدة البيانات كلما قل عدد ملفات سجل الإعادة المطلوبة للاستعادة وكلما زادت سرعة العملية.

إعداد سرية قاعدة البيانات

بالإضافة إلى التأكد من سرية البيانات بصورة فعلية، تتمثل المنطقة الأخرى التي يجب أن ننظر إليها في التأكد من مراقبة المستخدمين بصورة محكمة فيما يفعلونه وما لا يفعلونه. إذا كان بإمكان المستخدمون تغيير البيانات في حالة عدم تمتعهم بصلاحية الوصول إلى البيانات، فإن البيانات من الممكن معرضة للخطر أكثر من إمكانية فقدانها من خلال حدوث تلف في الأجهزة. من الممكن أن تكون عملية تشغيل الأعمال على بيانات غير صحيحة أسوء من عدم وجود بيانات على الإطلاق.

لحسن الحظ، يزودنا أوراكل مرة ثانية بمجموعة من الأدوات للسماح لنا بمراقبة سرية قاعدة البيانات إلى المستوى الذي نريده. حتى لو لم تكون المحلل الرئيسي لقاعدة البيانات، فإن عملية إعداد السرية سوف تكون عبارة عن مجهود مشترك بينك كمطور وبين مسئول قاعدة البيانات.

هناك بنود مفتاح أساسية نحتاج إلى بحثها:

- تكوين مستخدمين
- تخصيص امتيازات
- استخدام أدوار
- إعداد حصص على استخدام .جداول ملف الجداول
- استخدام المراجع لمراقبة استهلاك موارد النظام
- استخدام الإجراءات المخزونة لدعم السرية

تكوين مستخدمين

يشابه كون الشخص مستخدما لقاعدة بيانات مثل وجود حساب لشخص في بنك: حيث يصبح لديك اسما وكلمة مرور وبدونهما لا يمكنك عمل أي شيء. مع ذلك، حتى مع هذه الامتيازات، ينحصر ما يمكنك عمله حتى تود بعض النقود أو ترتب للحصول على مبلغ مسحوب: تحتاج إلى امتيازات إضافية لأية إمكانيات إضافية تريدها.

تنطبق نفس القواعد على قاعدة بيانات سرية. على الرغم من وجود قواعد بيانات مجموعة عمل على أساس الملفات مفتوحة لكل فرد بصورة افتراضية مع وجود قدرة محدودة لمراقبة إجراءات المستخدمين، لا يسمح أوراكل لك بالقيام بتنفيذ أي شيء بصورة افتراضية، ولا حتى تسجيل الدخول! يعتبر هذا النوع من السرية إلزاميا بالنسبة لقاعدة بيانات الإنتاج. لتكوين مستخدم في قاعدة بيانات، استخدم أمر CREATE USER كما هو موضح في المثال التالي:

```
CREATE USER nick
  IDENTIFIED BY ummagumma
  DEFAULT TABLESPACE tbs ;
  TEMPORARY TABLESPACE tbs_temp ;
```

في الواقع يعتبر هذا هو الحد الأدنى الواجب استخدامه. إذا لم يتم تحديد ملفات الجداول، فسيتم افتراضها إلى ملف جدول system، وهو من الأمور الواجب تجنبها بعناية. كما ذكر سابقا، لا يعطي هذا المستخدم الحق في الدخول، حتى لو كان بإمكانها، فليس لديها أي حق في استخدام أية مساحة. يمكنك داخل أمر CREATE USER إعطاء المستخدم حصة على ملفات الجداول-أي حجم مساحة القرص المسموح لهم باستخدامها داخل ملف جدول لتكوين وحدات قاعدة بيانات مثل الجداول والفهارس. يتم تنفيذ ذلك كما يلي:

```
CREATE USER nick
  IDENTIFIED BY ummagumma
  DEFAULT TABLESPACE tbs ;
  TEMPORARY TABLESPACE tbs_temp
  QUOTA 500K ON tbs ;
```

يمكنك تغيير سمات حساب المستخدم في وقت آخر باستخدام أمر ALTER USER، على سبيل المثال، لتغيير كلمة المرور وإضافة حصة إلى ملف جداول آخر ("أفلام")، ستستخدم

```
ALTER USER nick
  IDENTIFIED BY saucerful
  DEFAULT TABLESPACE movies ;
  TEMPORARY TABLESPACE movies_temp
  QUOTA 100K ON movies ;
```

تحذير يجب أن تتطلب معايير موقعك تعديل كلمات المرور بصفة دورية. بالنسبة للتجربة التي مررت بها، يكره جميع المستخدمون هذا الوضع، ولكن لا يجب أن يكون هذا مبررا لنسيان ذلك. تتمثل الفرص في أنك لن تلزم جميع المستخدمين بتشغيل جلسة عمل SQL لتغيير كلمات المرور، وهو الأمر الذي لا يمكنك تنفيذه بسهولة من إجراء مخزون. سوف يكون هذه الأمر مجرد عملية يدوية بالنسبة لمسئول قاعدة البيانات.

تلميح لا تفترض مباشرة وجوب منح المستخدم أية حصة على الإطلاق. إذا كان لديك قاعدة بيانات مطورة مع تطبيق طرف أمامي، لا يحتاج المستخدمون الذين يستخدمون التطبيق أية حصة على ملفات الجداول. يتم فرض الحصة فقط لتكوين الجداول والفهارس الخ وكذا الكثير من الوقت المبذول. لا يجب على المستخدمين النهائيين القيام بذلك. في حالة الاحتياج إلى تكوين جدول مؤقت، من الممكن تنفيذ ذلك من داخل إجراء مخزون مملوك في مخطط آخر.

أخيرا، لحذف مستخدم من قاعدة بيانات، وكذا كل الوحدات من المخطط، استخدم أمر DROP USER "إسقاط مستخدم" من خيار CASCADE "تتالي" كما يلي:

```
DROP USER video_user CASCADE ;
```

يظهر هذا الموقف في معظم الأحيان عند انتقال المستخدم إلى وظيفة أخرى، سواء داخل الشركة أو خارجها ولا يجب الاستخفاف بهذا الأمر. ولكن يعتبر تنفيذ هذه المهمة واحدا من الأمور المتروكة حتى يتوافر لديك الوقت ومن الممكن نسيانه والتسبب في مشكلة سرية محتملة. حاول الحصول على إخطار رسمي من مكتب الأفراد وأعطه أولية قصوى لإسقاط المستخدم بأقصى سرعة ممكنة.

عاذا يتم تسمية مستخدم

يجب استخدام أية معايير مواقع موجودة. في حالة عدم وجود أي معيار استخدم حينئذ المعايير الشائعة مثل الحرف الأول من الإسم الأول زائد الإسم الأخير، على سبيل المثال "معلي" بالنسبة لإسمي أنا شخصيا. إذا لم يزل لديك مستخدمين بنفس الاسم، يجب عليك إضافة الحرف الأول من اسم الأب

استخدام صحة نظام التشغيل

في بيئة شبكية، يتم الدخول بصورة نموذجية إلى الشبكة قبل تشغيل تطبيق مستخدم نهائي. باستخدام صحة نظام التشغيل، يقوم أوراكل بفحص ما إذا كنت قد قمت بالدخول إلى الشبكة، إذا كان قد تم ذلك وكان اسم المستخدم الخاص بك هو اسم مستخدم أوراكل صحيح، فلست بحاجة إلى إعطاء كلمة مرور. يعتبر هذا الأمر رائعا من وجهة نظر إدارية في أنه لا يجب إدارة كلمات مرور لمستخدمي أوراكل حيث أن كل فرد صالح للدخول بواسطة تسجيل الدخول إلى الشبكة. مع ذلك، فإن هدفك كمطور للفيجوال بيسيك يتمثل في تكوين تطبيقات مستخدم نهائي يتم تشغيله في طور المزود الخادم. باستخدام صحة نظام التشغيل، من الممكن أن يقوم أي شخص على دراية بما يفعله، بتكوين اسم مستخدم يعطيهم صلاحية الوصول إلى قاعدة البيانات آليا.

تحذير : تعتبر عملية صحة نظام التشغيل غير آمنة للاستخدام مع أوراكل في معظم الأحوال. استخدم صحة كلمة المرور لأفراد مع تغيير كلمات المرور بصفة منتظمة.

عند الإصرار على استخدام صحة نظام التشغيل، لا تعتبر العملية صعبة. يتم إضافة معامل بدء os_authent_prefix إلى بداية اسم مستخدم نظام التشغيل لإنتاج اسم مستخدم أوراكل. السلسلة الافتراضية لهذا المعامل هي "OPS$"، ولكن يمكنك إعدادها على ما تريده حتى لو كان فارغا.

على سبيل المثال، إذا كنت تقوم بتشغيل أوراكل على ويندوز NT وكانت لديك شبكة NT، وكان اسم الدخول إلى الشبكة الخاص بك هو "معلي" وكان لديك معامل بدأ افتراضي للبادئة "OPS$"، قم بتكوين حساب مستخدم أوراكل جديد كما يلي:

```
CREATE USER OPS$bmills IDENTIFIED EXTERNALLY ;
```

تبين كلمة المفتاح خارجيا، بدلا من كلمة المرور، بأنه سيتم توثيق مستخدم قاعدة البيانات بواسطة نظام التشغيل. في حالة إعداد المعامل على خالي، سيكون اسم مستخدم نظام التشغيل هو نفس اسم مستخدم أوراكل.

تلميح توفر صحة نظام التشغيل صلاحية الوصول العادية إلى قاعدة البيانات. وليس صلاحية وصول SYSDBA (أي امتيازات بدء التشغيل والإغلاق). إذا أردت ممارسة الإدارة عن بعد، ستحتاج إلى إعداد ملف كلمة مرور.

تخصيص امتيازات

الآن قمت بإعداد زمرة مستخدمين يحتاجون إلى الوصول إلى التطبيق. في هذا الوقت، لديهم حساب فقط: ولا يمكنهم القيام بأي شيء في أوراكل ولا حتى الدخول إلى قاعدة البيانات. كمسئول قاعدة بيانات، يجب أن تقوم صراحة بمنح امتيازات إلى المستخدمين للسماح لهم بالوصول إلى وحدات قاعدة البيانات. يمكنك الحصول على القدر الذي تريده من المراقبة بالنسبة لعملية منح الامتيازات. تنقسم أنواع الامتيازات التي يمكن منحها إلى المستخدمين إلى نوعين منفصلين هما: امتيازات النظام وامتيازات الوحدة.

امتيازات النظام

النوع الأول من الامتيازات هو امتياز النظام. تحتاج إلى امتيازات النظام لتنفيذ جمل SQL معينة بدلا من الوصول إلى وحدات خاصة. كمثال على ذلك، هو الحق على إسقاط الجداول في أي مخطط في قاعدة البيانات. هناك عدد كبير من ميزات النظام وهي ما سيمنحه مسئول قاعدة البيانات إلى مطور. سوف نغطي بعضا من هذه الامتيازات الهامة من وجهة نظر مطور في الفصل ٦ "تنفيذ قاعدة بيانات".

امتيازات الوحدة

النوع الثاني هو امتيازات الوحدة. تنطبق امتيازات الوحدة بصفة خاصة على نوع إجراء واحد في وحدة معينة. مثال ذلك، إذا أردت أن يتوفر لمستخدم الحق في اختيار البيانات من جدول، قم بمنحه صلاحية وصول SELECT إلى ذلك الجدول. لا يعطيه ذلك الحق في تغيير أي شيء في ذلك الجدول أو استعرض أي شيء في أي جدول آخر. يتم منح لمتيازات الوحدات بصورة نموذجية من جانبك كمطور إلى المستخدمين النهائيين. بإمكان مالك الوحدة منح صلاحية الوصول إلى تلك الوحدة.

هناك بعض امتيازات وحدات قليلة فقط، وسيتم مناقشتها في الفصل ٦. تعتبر امتيازات الوحدات من الطرق الهامة لتأمين الوصول إلى بياناتك، وكما سنرى في كل أنحاء هذا الكتاب، يوف تشكل تأثير رئيسي على تصميم التطبيق.

منح الامتيازات

لمنح امتيازات النظام إلى المستخدمين، تحتاج أولا إلى القدرة على التمرير داخل هذه الامتيازات. لا توجد أية امتيازات معينة مطلوبة للسماح لمسئول النظام لمنح امتيازات النظام إلى المطورين أو للسماح للمطورين بمنح امتيازات الوحدة على مخططهم إلى المستخدمين النهائيين، ولكن إذا أردت القيام بأكثر من ذلك، يجب قراءة الأقسام الخاصة بأمر في دليل مرجع SQL لأوراكل.

يأخذ أمر GRANT SYSTEM PRIVILEGE "منح امتيازات النظام" الشكل التالي:

```
GRANT privilege TO user_name ;
```

إذا أردت قيام امتيازات النظام بتكوين جداول في أي مخطط، يجب استخدام

```
GRANT CREATE ANY TABLE TO jwoods ;
```

يعتبر أمر GRANT OBJECT PRIVILEGE "منح امتياز النظام" مماثلا ولكنه يتضمن اسم الوحدة الذي يشير إليها الامتياز:

```
GRANT privilege ON object_name TO user_name ;
```

لذا إذا أردت منح امتياز الوحدة اختيار أية بيانات من جدول معين، يمكنك استخدام

```
GRANT SELECT ON video_store.movies TO jwoods ;
```

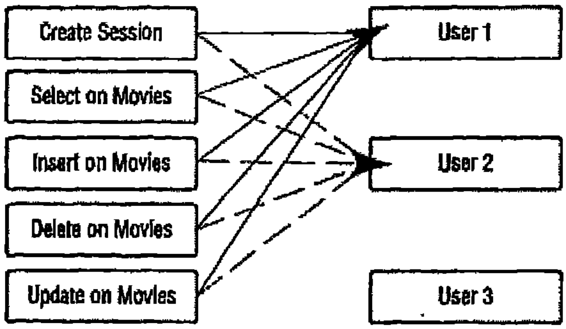
معه أي من هذه الأوامر، يمكنك تضمين أكثر من امتياز في المرة الواحدة، لإعطاء المستخدم امتياز القيام بأي شيء على البيانات في جدول، يمكنك استخدام

```
GRANT SELECT, INSERT, UPDATE ON video_store.movies TO jwoods ;
```

ملاحظة تسمح امتيازات وحدة معينة، مثل INSERT "إدراج" وUPDATE "تحديث" بوجود قائمة من الأعمدة التي يمكن أن تتأثر بالامتياز. مثال ذلك، يسمح "GRANT UPDATE (movie_name) ON movies TO jwoods" التي تخص jwoods فقط بإدراج صف بقيمة معرف فقط.

يعطيك الشكل ٤-٧ بفكرة عن عدد الأوامر التي تحتاجها لإعداد المستخدمين. بالنسبة لكل مستخدم، يجب المرور عبر نفس العملية. من الممكن وضع ذلك داخل سكريبت، والذي يعتبر مناسبا للإعداد في المقام الأول. ولكن ما زال هذا يعتبر من كوابيس الصيانة كما هو الحال عند إضافة جدول جديد إلى التطبيق: يجب العثور على كل المستخدمين الذين لديهم صلاحية وصول إلى هذا الجدول الجديد ثم إضافتهم جميعا.

الشكل ٤-٧ منح امتيازات للمستخدمين.



استخدام الأدوار

على الرغم من أنه من الممكن من الناحية العملية إدارة كل نظام وامتياز وحدة في قاعدة البيانات بإعطاء منح إلى المستخدمين الفرديين، من الممكن أن تصبح هذه الطريقة غير قابلة للإدارة بسرعة بمجرد توافر عدد ضئيل من المستخدمين. إذا كان عليك مراجعة الحقوق المعطاة إلى الأفراد، فمن المحتمل أنك ستلاحظ انه بإمكانك فرز الامتيازات داخل مجموعتين منفصلتين.

على سبيل المثال، هناك ثلاثة أنواع منفصلة من التطبيقات، مثل الموارد البشرية والهندسية والإنتاج، ومن المحتمل أنه هناك اكثر من ثلاثة أنواع من صلاحية الوصول إلى البيانات التي يحتاجها الأفراد: ستحتاج مجموعة إلى تحديث بيانات الموارد البشرية بينما سيتم السماح للبعض باستعراض جزء منها. لترتيب هذه الامتيازات داخل زمر محسوسة وممكن إداراتها، يمكننا استخدام مفهوم الدور.

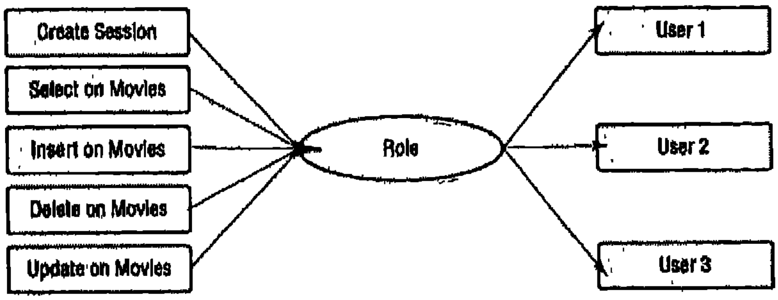
الدور هو طريقة تحصيل امتيازات عديدة داخل زمرة مفردة. عند منح هذا الدور إلى مستخدم مفرد، يتم حصولهم على كل الامتيازات المرتبطة بهذا الدور (راجع الشكل ٤-٨). باستخدام اختيار دقيق للأدوار، يجب أن تكون قادرا على إعداد الأشياء التي يحتاجها كل مستخدم لكي يتم منحها ثلاثة أدوار فقط على الأكثر.

تلميح: يعتبر استخدام الأدوار أكثر كفاية من منح الامتيازات الفردية ولهذا يجب التفكير بعناية مع وجود سبب محدد في حالة تقرير منح الامتيازات إلى الأفراد.

عند إعداد الأدوار إلى المطورين، تعتبر امتيازات النظام التي تريد منحها إلى الدور هي كالآتي:

- تكوين جدول
- تكوين مسلسل
- تكوين إجراء
- تكوين زناد
- تكوين ملف جداول
- تكوين جدول اعتباري

سوف تعتمد ميزات وحدة أدوار المستخدم النهائي بصورة واضحة على أية جداول تم إعدادها وإذا ما كان قد تم إعطاء المستخدمين إمكانية الوصول إلى الجداول من خلال الإجراءات المخزونة أو باستخدام ميزات التحديث.



الشكل ٤-٨ تخصيص امتيازات للأدوار.

ملاحظة بالإضافة إلى إمكانية تخصيص أدوار إلى مستخدمين، يتملك أوراكل أيضا مفهوم المرجع، والذي يبين حدود موارد المستخدم مثل الحد الأقصى وقت وحدة المعالجة المركزية والحد الأقصى لوقت التوقف والحد الأقصى لمرات القراءة المسموح بها. يجب استخدام هذه القيود إذا كانت أنواع الاستعلامات والحركات الموضوعة في قاعدة البيانات سوف تتسبب في عجز الموارد.

الأدوار المحددة مسبقا

من الممكن تسهيل العمل الخاص بك بواسطة استخدام الأدوار المحددة مسبقا. هذه الأدوار عبارة عن أدوار قليلة يمكنك منحها بمر واحد بسيط. هذه الأدوار معروضة في الجدول ٤-٢.

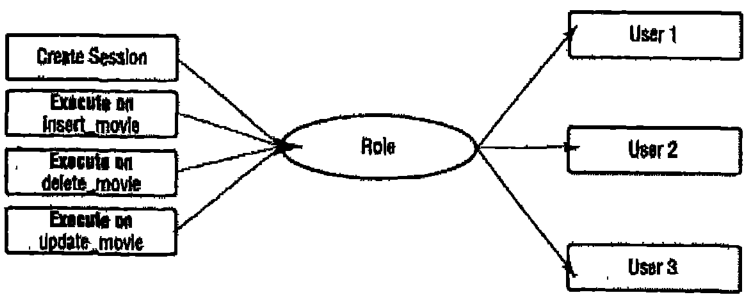
الجدول ٤-٢: الأدوار المحددة مسبقا الممكن استخدامها

الدور	البيان
التوصيل	امتيازات الصيانة
المورد	امتيازات المطور
مدير قاعدة البيانات	امتيازات مدير قاعدة البيات الكاملة
EXP_FULL_DATABASE	تصدير قاعدة البيانات بالكامل
IMP_FULL_DATABASE	استيراد قاعدة البيانات بالكامل

في التركيب الأوسع، من الممكن أن تريد استخدام دور مدير قاعدة البيانات لنه من الممكن أن تريد مراقبة المهام التي يقوم بها كل شخص. على سبيل المثال، من الممكن أن تريد قيام شخص ما بمسئولية القيام بهام النسخ الاحتياطية وآخر بالموالفة. في هذه الحالة، سوف تضطر إلى تكوين أدوار خاصة بك تحتوي على مجموعة فرعية من ميزات مدير قاعدة البيانات.

الإجراءات المخزونة لدعم السرية

عندما تقوم بمنح الامتيازات، يمكنك مراقبة أي مستخدمين مسموح لهم بقراءة أو تحديث جداول محددة أو أعمدة محددة. مع ذلك، هذا هو مستوى أساسي من السرية. مما لا شك فيه أن هذا يعتبر من الضروريات الأساسية، ولكن من الواجب أن تقوم قاعدة البيانات المحسنة بفرض مستوى أفضل من المراقبة. من الممكن مراقبة من يقوم بتحديث البيانات دون طريقة التحديث. يجب تقديم هذا بواسطة تنفيذ قواعد الأعمال.



الشكل ٤-٩ تخصيص امتيازات إجراءات مخزونة للأدوار.

تحتاج قواعد الأعمال إلى تغليف الوصول إلى البيانات الخام. تتضمن الأساليب الحالية استخدام مكونات برامج الطبقة المتوسطة بين العميل وملقم قاعدة البيانات. مع ذلك فإن أسهل الطرق لتنفيذ قواعد الأعمال هو استخدام الإجراءات المخزونة. الإجراءات المخزونة هي عبارة عن الاسم المعطى إلى تعليمات البرمجة التي تم تخزينها في قاعدة البيانات، ويمكنك التأكد من أن الطريقة الوحيدة لتغيير أية بيانات هو من خلال هذه الإجراءات. سو نناقش هذا بتفصيل اكثر في الفصل ١٠، تكوين إجراءات مخزونة. أما الآن سوف نركز فقط على احتياجك لتنفيذ امتياز استخدام الإجراء المخزون، وكما هو ملاحظ مما سبق فإن أفضل الطرق لتنفيذ ذلك هو من خلال استخدام الدور، انظر الشكل ٤-٩.

مهام الصيانة الدورية

من الواجب الاحتفاظ بقائمة من مهام الصيانة المحددة التي يتم تنفيذها على أساس دوري. سوف يكون لدى كل مدير قاعدة بيانات المهام المفضلة لديه، ولكن ما يلي من الممكن أن يكون من الأمور المفيدة كأساس لقائمة المهام الفردية.

يوميا

- قم بفحص أية وظائف إجمالية مفاجئة للتأكد من تشغيلها في الوقت المحدد وبدون أخطاء.
- قم بفحص تصدير قاعدة البيانات المفاجئة.
- قم بفحص النسخ الاحتياطي السريع.
- راجع سجلات التنبيه لمعرفة الأخطاء.
- قم بمراقبة عمليات أوراكل لمعرفة أوجه القصور الموجودة في الموارد.
- قم بمراقبة المساحة المتروكة في ملفات البيانات.
- قم بمراقبة الجداول التي من الممكن أن تكون غير قادرة على النمو داخل ملف بيانات.
- قم بمراقبة أداء وظائف المجموعة المفاجئة.

أسبوعيا

- قم بفحص أحجام ملفات السجلات.
- قم بمراقبة ملفات البيانات السريعة لمعرفة المساحة الفارغة والموقع الصحيح.

- قم بفحص حجم المجلدات للحصول على الناتج خلال الأسبوع. بالنسبة لقواعد البيانات ذات نشاط حركة مكثف، من الممكن أن يكون هذا مهمة يومية.
- قم بعمل تدقيق سرية ومستخدم.

شهريا

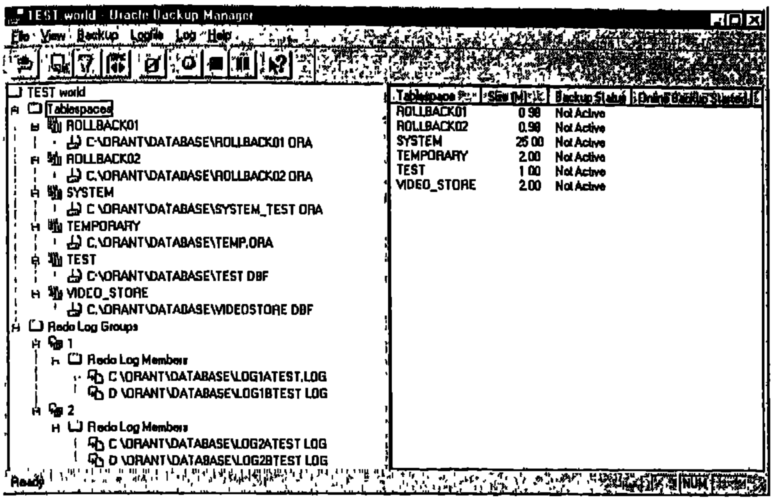
- قم بمراجعة إحصائيات مستوى أداء قاعدة البيانات.
- قم بفحص تجزئة وتسلسل الامتدادات.
- قم بفحص نمو ملف الجدول وتعديل حجم الملف الأساسي.

استخدام مدير المؤسسة

لقد شاهدنا أوامر SQL المطلوبة لإدارة قاعدة البيانات في هذا الفصل حتى الآن. من الأمور الأساسية بالنسبة لمدير قاعدة البيانات المحترف الحصول على فهم شامل لهذه الأوامر. مع ذلك فكما هو الحال بالنسبة لمعظم نواحي تصميم البرامج، تأتي الآن حزم برامج الوسيط الرسومي، والتي تسمى مدير المؤسسة لأوراكل، مرفقة مع إصدار المؤسسة لأوراكل. تضع البرامج واجهة وسيط رسومي في كل أوامر SQL التي شاهدناها في هذا الفصل. تتمثل ميزة مدير المؤسسة لأوراكل في توافر وقت أيسر لدى المستخدم غير الدائم للقيام بمهام إدارية بعض الوقت. أم الجانب السيئ فيتمثل في نسيان الطرق اليدوية كما هو الحال بالنسبة لاستخدام الآلة الحاسبة الصغيرة. أحيانا ما تكون هذه الطرق اليدوية من الأمور التي لا بد منها.

يعطيك مدير المؤسسة إمكانية النقر والتأشير لعرض معلومات إجمالية حول قاعدة البيانات وكذا السماح لك بالحصول على عرض تفصيلي للمعلومات الهامة. بدلا من عرض بيان كامل لمدير المؤسسة، سوف نقوم بزيارة قصيرة لربع من الإمكانيات الرئيسية التي يقدمها لك المنتج. يجب أن يكون هذا كافيا لإعطائك فكرة عن إمكانيات البرنامج ومن المحتمل أن يحفزك على شراءه.

أول تطبيق داخل مدير المؤسسة والذي سوف نقوم بإلقاء نظرة عليه هو مدير النسخ الاحتياطي (انظر الشكل ٤-١٠). حيث يعرض لك نظرة عامة حول ملفات الجداول وزمر سجلات إعادة التشغيل مع السماح لك بمراقبة النسخ الاحتياطية.



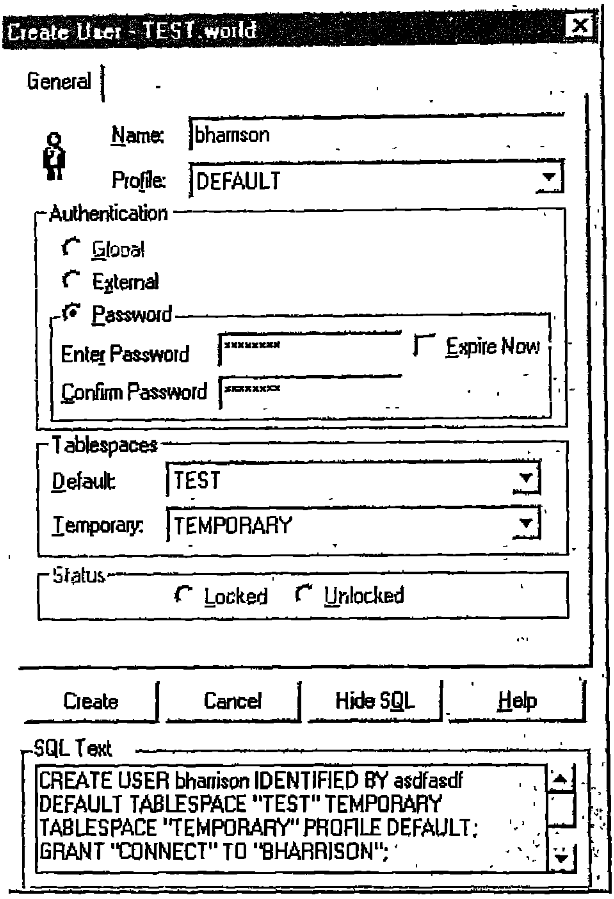
الشكل ٤-١٠

تطبيق مدير النسخ الاحتياطي.

ملاحظة يتمثل جمال مدير المؤسسة في إمكانية إعداده بحيث يمكنك إدارة كل قواعد البيانات على الشبكة من لوحة تحكم واحدة.

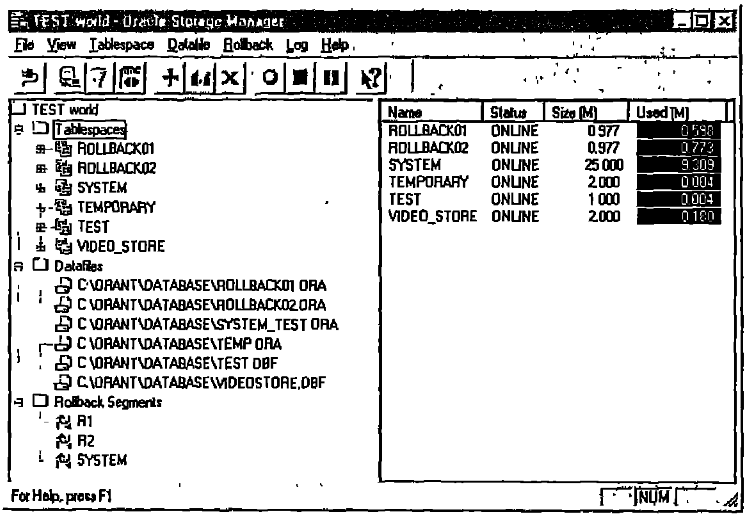
التطبيق التالي هو مدير السرية. حيث يعرض لك كل المستخدمين والأدوار في قاعدة البيانات. كما يسمح لك مدير السرية باستعراض أية أدوار تم تخصيصها إلى أية مستخدمين وأية امتيازات تم تخصيصها إلى أية أدوار ومستخدمين. تعتبر من الميزات الهامة، ولكنها تقوم بأكثر من ذلك. حيث يمكنك بسهولة تكوين مستخدمين جدد (انظر مربع حوار تكوين مستخدم الموجود في الشكل ٤-١١) مع تخصيص امتيازات جديدة إلى المستخدمين. بواسطة النقر على زر عرض SQL، يمكنك مشاهدة أمر SQL الكامل والذي يقوم بتحقيق نفس التأثير من موجه سطر الأوامر.

يسمح لك مدير التخزين بعرض التخزين المأخوذ بواسطة ملفات الجداول وملفات البيانات و مقاطع إلغاء التعديلات. بالنسبة لطريقة العرض المفضلة لدي هي تلك المعروضة في الشكل ٤-١٢ والتي تسمح لك، بنظرة واحدة، مشاهدة ماهية ملفات الجداول بالكامل.



الشكل ٤-١١

تطبيـــق مديـــــر السرية.



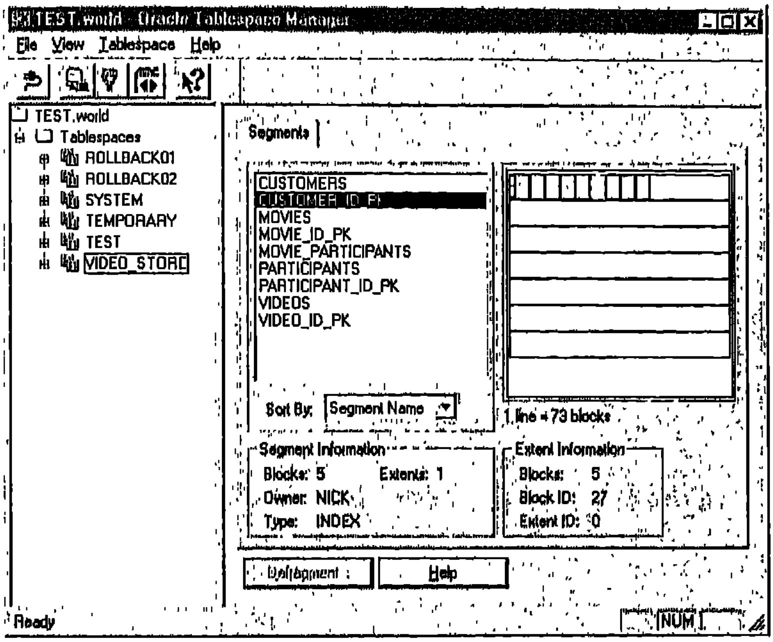
الشكل ٤-١٢

تطبيـــق مديـــــر التخزين.

المثال الأخير لمدير المؤسسة في الواقع هو الإضافة. (يجب أن يتم الدفع لهذه الميزة بصورة منفصلة). يسمى حزم الأداء ويحتوي، بصورة ملحوظة، على نظام خبير لتحليل أداء قاعدة البيانات. هذا هو التطبيق الذي يمكنك من موالفة قاعدة البيانات دون الحاجة إلى سنوات من الخبرة في الموالفة. لا يجب الاستخفاف بعملية الموالفة، حيث تحتاج إلى فهم جيد لما يوصي به نظام الخبير قبل التغيير المرفوض لمعاملات النظام.

أحد جوانب حزم الأداء التي أستخدمها أنا شخصيا هي مدير ملف الجداول. حيث يسمح لك بالتعمق داخل ملف الجداول والعرض الرسومي لتوزيع (ومن المحتمل التجزئة) الجداول داخل المساحة (انظر الشكل ٤-١٣). تسمح لك الميزات الأخرى بالقيام بتنفيذ الإدارة الآلية. تعتبر حزم الأداء من الإضافات العظيمة لمجموعة أدوات مدير قاعدة البيانات المشغول.

الشكل ٤-١٣
مدير ملفات الجداول في مجموعة الأداء الاختياري



خلاصة

لقد قدم هذا الفصل مقدمة لإدارة قاعدة بيانات أوراكل. كما غطى منطقتين رئيستين: استراتيجيات وسرية النسخ الاحتياطي لقاعدة البيانات. تحتاج إلى إعطاء اهتمام خاص لكل من هاتين المنطقتين للحصول على مستوى جيد من الثقة في سرية وجدارة البيانات. إذا أصبح هذا جزءا من وظيفتك، يجب الحصول على مستوى عالي من التدريب قدر الإمكان. في هذه الحالة، لن تحتاج فقط إلى فهم شامل لكل الأدوات المتاحة فقط ولكن أيضا الحصول على خبرة في مناطق مثل الموالفة.

يعتبر هذا الفصل والفصلين السابقين من الأمور الكافية لتقديم الأساس الذي يتم عليه بناء التطبيقات. إذا كنت تقوم بالعمل مع مدير قاعدة بيانات خبير يجب أن تتوافر لديك المعرفة الكافية لفهم أية مهام يتم تنفيذها وكذا لتقديم المساعدة الممكنة. ومن المحتمل أن تكتشف بعض المناطق التي من الممكن تحسينها. من الممكن أن نعمل الآن تجاه بناء قواعد بيانات ذات جداول وإجراءات مخزونة الخ، مع إمكانية الوصول من خلال برامج فيجوال بيسك.

٥

تصميم قاعدة البيانات

عند بدء تكوين قواعد البيانات للمرة الأولى، قد توجد لديك رغبة تجاهل الطرق الشكلية أثناء مرحلة التصميم. وهذا حقيقي خصوصاً عند تطوير التطبيقات الصغيرة التي يوجد بها مخزن بيانات محدود متعلق بها أو إذا كنت مطور لا يمكنك الانتظار حتى تصل إلى مرحلة * applications. وعلى قدر ما تكون قاعدة البيانات أصغر على قدر ما يكون التصميم أقل أهمية. ومع ذلك، فإن العكس صحيح. وكلما تطورت قاعدة البيانات، تبدأ الاختصارات التي قمت بتجميعها معاً في الظهور بشكل أوضح. وفي أفضل الأحوال، تكون هذه التطبيقات مذهلة، وفي أسوأ الأحوال، تكون بطيئة ولا يمكن الاعتماد على البيانات الموجودة بها. وفي النهاية، سوف تجد أنك قمت بالمهمة بالشكل المناسب وسوف تستمر قاعدة البيانات في التطور وإمكانية استخدامها في طرق أكثر مما تحلم به.

يعتبر تصميم قاعدة البيانات مرحلة واحدة لا يمكن تجزئتها في جملة عملية تصميم النظام. ويعتمد الجزء الذي يتم استخدامه على البيئة التي تعمل بها. فمن ناحية، قد تكون مسئولا عن كل شيء بالكامل وهذه تعد مهمة غير مشجعة: فتوجد أشياء كثيرة لابد أن تتعلمها وسوف تواجه أشياء ليس لك دراية بها أيضا. ومن الناحية الأخرى، قد تكون تعمل في مكان مهيأ ومحكم مع مجموعات منفصلة لتصميم وصيانة وتنفيذ قاعدة بيانات بغض النظر عن التطبيقات الأمامية والطرفية. وفي مثل هذا الموقف، من الأفضل فهم أكبر قدر ممكن عن العملية.

ومن الممكن تقسيم عملية تصميم قاعدة البيانات إلى ثلاث مراحل مميزة:

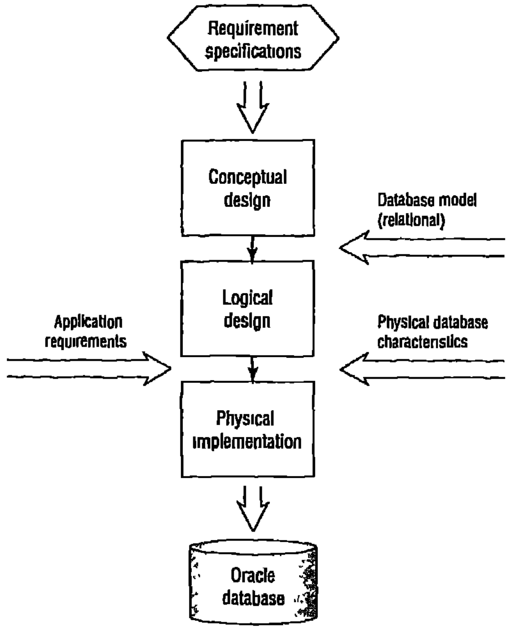
التصميم المفاهيمي: يأخذ كل خصائص المتطلبات التي حصلت عليها نتيجة التفاعل مع مجتمعات المستخدمين ويقوم بتكوين التصميم المستقل عن أي شئون برامج أو معدات. ويعتبر مخطط كيان العلاقة اكثر تقنية شائعة في هذه الحالة.

التصميم الاعتباري: ويقوم بتخطيط النموذج المفاهيمي إلى نموذج اعتباري لنوع نظام إدارة قاعدة البيانات الذي نقوم باستخدامه. وما نريد الحصول عليه هو التأكد أن التصميم سليم فنيا وبه احتياطات صيانة وحد أدنى من البيانات زائد عن الحاجة.

التنفيذ الحقيقي: ويقوم بتنفيذ جداول البيانات الحقيقية وعمل قرارات التصميم لتدعيم الفائدة الناتجة عن قاعدة البيانات مع التطبيقات المطلوبة. وتعتبر هذه المرحلة النهائية في تكوين قاعدة البيانات.

سوف تجد أن تفاصيل هذه المراحل تختلف بناءا على الأسلوب المستخدم، ولكن تتبع هذه التفاصيل الإطار العام الذي تم شرحه أعلاه والموضح في الشكل ٥-١ بشكل عام.

الشكل ٥-١
عمليـــة تكويـــن قاعدة البيانات.



حاول أن تضع هذه المراحل في اعتبارك أثناء تطوير التصميم. حتى عند إجراء هذه المراحل لمرات عديدة، يصبح من السهل نسيان جزء أو أكثر، وسوف يعود هذا عليك بالمتاعب ثانية.

ومع أن المراحل مميزة، فليس من اللازم أن يكونوا على وتيرة واحدة. على سبيل المثال، إذا كنت تعرف خاصية محددة عن قاعدة البيانات تؤثر على التصميم، يمكنك استخدام هذه المعلومة في مرحلة التصميم الاعتباري. وسوف تظهر نتيجة ذلك طبيعيا كلما نمت التجربة.

وعلى مدار هذا الفصل، سوف نقوم بتغطية التصميم المفاهيمي وكذلك التصميم الاعتباري بتفاصيل اكثر. كما أننا سوف نتعرض للتنفيذ الحقيقي في الفصل ٦ عندما نأخذ التصميم المطور في المرحلتين الأولى والثانية ونقوم بتكوين قاعدة بيانات أوراكل التي تنقل كل ما جئنا به.

التصميم المفاهيمي

لإنتاج تصميم قاعدة بيانات جيد، نستخدم عملية تسمى تخطيط نماذج البيانات. ونموذج البيانات هو عبارة عن تمثيل موجز لبيئة حقيقية تحتوي على وحدات وعلاقتهم. ويمكنك إجراء تخطيط نماذج أعمال لمخزن فرعي أو اقتصاد بلد لإغراض الموازنة وهي نفس العملية مع اختلاف في درجة التعقيد. والغرض من هذا النموذج هو تمثيل تلك البيئة بدون النظر إلى تقنية قاعدة البيانات التي تستخدمها أو التطبيقات التي يمكن تشغيلها في المقابل لها.

وفي المرحلة المتعلقة بالمفهوم، سوف تقوم بتكوين نموذج البيئة. وعليك جمع معلومات بقدر الإمكان من المستخدمين بخصوص النظام الحالي والنظام المطلوب، ولكن لا تركز على التطبيقات الحالية كثيرا سواء كانوا على الكمبيوتر أو على ورق. ولكن بالأحرى، عليك التركيز على اكتشاف العمليات الرئيسية وأي تدفق بيانات يناسب النظام. ثم يمكنك بعد ذلك تشكيل عناصر البيانات بالضبط، والأسلوب العام كالتالي:

- تحديد متطلبات البيانات.
- تجهيز الرسومات التخطيطية لتدفق البيانات.
- إنتاج تخطيط لحالة الانتقال إذا كانت قابلة للتطبيق.
- توثيق البيانات في تخطيط كيان العلاقة.
- تكملة التخطيط بواسطة سمات الكيان.

والأهم في هذا كله بالنسبة لهذا الفصل هما الخطوتين النهائيتين. تقوم هذه الخطوات بتوثيق كل عناصر البيانات التي لم يتم تغطيتها من ناحية مطابقتها مع تكوين قاعدة البيانات. وإذا لم تكن معتاد على الخطوات الأخرى،عليك الرجوع إلى واحد من الكتب الجيدة على تحليل النظم.

الرسومات التخطيطية لكيان العلاقة

تعتبر الرسومات التخطيطية لكيان العلاقة الطريقة الأفضل لعرض العلاقات بين الكيانات. توجد كثير من الطرق المختلفة لتكوين الرسومات التخطيطية هذه. وسوف نقوم في هذا الفصل بمراجعة ما المقصود من الرسومات التخطيطية هذه في شكلهم الأساسي.

يمكن اعتبار الكيان طبقة من الوحدات الحقيقية. وقد يكونوا على سبيل المثال وظائف أو أشخاص أو مدن. فلابد أن تكون قادر على تعريف كل كيان بشكل منفرد داخل السياق الذي تعمل فيه. ويمكنك إجراء ذلك بواسطة اسم أو رقم سري أو أي عنصر رئيسي آخر. يمكنك بعد ذلك تحديد السمات لكل واحد من الكيانات. وتعتبر السمات معلومات إضافية للإلمام بالكيانات مثل المسمى الوظيفي أو اسم الشخص. وفي الفيجوال بيسك، سوف تكون العملية قابلة للمقارنة لتعريف الطبقات، كل طبقة مع خصائصها التي سوف تظهر في النهاية في شكل صفوف وأعمدة في تصميم قاعدة البيانات الاعتباري.

وكمثال على تصميم قاعدة البيانات، سوف استخدم مثال بسيط ومعتاد وهو مخزن إيجار الفيديو، حيث قرر المالكين حفظ كل المعلومات عن الأفلام المؤجرة على الكمبيوتر في المخزن الخاص بهم بحيث يتمكن العملاء من البحث عن أي حقائق بسرعة تكون متاحة. ويقوم المستخدمون حتى الآن باستخدام الجداول الإلكترونية للتتبع الفيديو. وعلى الرغم أن الجداول

الإلكترونية كانت أساساً الحل الأرخص، إلا ثبت أنهم يسببوا قلق صيانة واستحالة بنـــاء تطبيـق على هذه الجداول.

والقصد هو إيجاد موقف واقعي بدون تعقيد. وكل ما يخدم به هذا التعقيد هو تعتيم إجـــراءات التصميم. وما هو مدهش أن العديد من الأعمال تستخدم الجداول الإلكترونيـــة كقـاعدة بيانـــات رئيسية لهم.

وفي هذا المثال، بعد مراجعة متطلبات الأعمال، يمكنك تسجيل الكيانات المختلفة التي تجدهـا هامة، مثل:

- الأفلام التي تريد تخزينها.
- نسخ الفيديو الفعلية من الأفلام.
- الممثلون.
- المخرجون.
- العملاء.

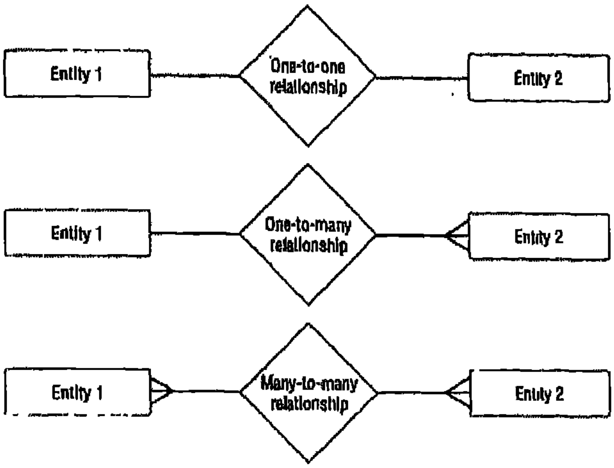
وعن الإتيان بالكيانات إلى البيئة، تقوم بعد ذلك بتحديد العلاقات الممكنة بينهم، مثل:

- الممثلون الأبطال في الأفلام.
- مخرجي تنفيذ الأفلام.
- وجود نسخة أو أكثر من كل فلم بالمخزن.
- إمكانية تأجير كل نسخة لعميل واحد.

تلميح تذكر أنه يمكن وصف الكيانات باسم والعلاقات بفعل.

ويعتبر الهيكل القياسي لتخطيط النموذج المفاهيمي هو استخدام المستطيلات للكيانات ومعينات للعلاقات. وعند إجراء ذلك، يمكنك رسم الخطوط بين المربعات لتحديد موقع تطابق العلاقـــات. وأخيراً، عليك تعريف عدد الوحدات لكل كيان مستخدم في كل علاقة. على سبيل المثـــال، مـن الممكن أن يشترك العديد من الممثلين في أفلام عديدة مختلفة، ولكن يمكن لعميل واحـــد إيجـار نسخة محددة من الأفلام على حدة. وتسمى هذه العملية بأصل العلاقات، وهي موضحة في الشكل ٥-٢.

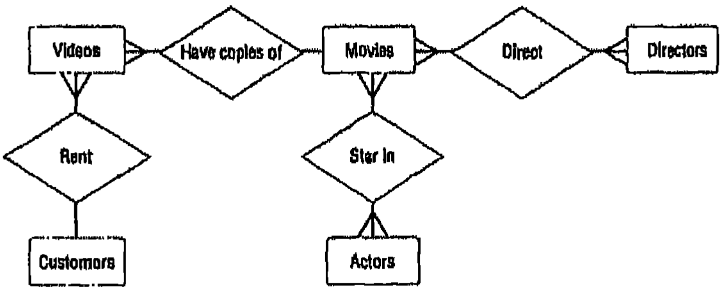
الشكل ٥-٢
بيان أصل العلاقات.



وعند تطبيق أسلوب التخطيط على قاعدة البيانات Movie قد تخرج بتصميم مثل الموضح في الشكل ٥-٣. واسمح لنا بالنظر إلى العلاقات بين الكيانات. ولابد أن تتطابق هذه العلاقات مع قواعد الأعمال الناتجة عن التفاعل مع المستخدمين في خواص المتطلبات. ويعتبر توثيق العلاقات في هذه الطريقة أداة ممتازة للتوصل إلى التصميم المتطور مع مستخدمين في النهاية. يوضح الشكل ٥-٣ العلاقات المرقمة التالية:

١- يمكن لكل عميل إيجار أفلام عديدة.
٢- من الممكن وجود نسخ حقيقية متعددة (أفلام) من الأفلام.
٣- من الممكن وجود ممثلين متعددين في فيلم، ويكون كل ممثل نجم في أفلام متعددة.
٤- من الممكن وجود مخرجين متعددين في فيلم وكل مخرج يمكنه الإخراج في أفلام عديدة.

الشكل ٥-٣
مخطط كيان علاقة قاعدة البيانات Movie.



إضافة السمات

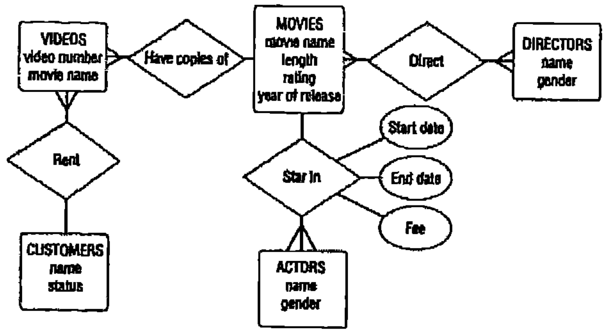
والآن قد تم تخطيط الكيانات الخاصة وحان الوقت لإضافة السمات لكل واحد. ويمكنك تخيل السمات كعناصر تصف الوحدات بتفاصيل أكثر. وعليك التفكير حول كل المعلومات الممكنة التي

قد تكون بحاجة إليها للإلمام بأي كيان، مثل الفيلم: فقد تريد معرفة اسمه وترتيبه وطوله وربما السنة التي صدر فيها. وقد تكون بحاجة كذلك لمعرفة النجم أو النجمة في كل فيلم، ولكن سوف يتم تمثيل هذه المعلومات بواسطة العلاقة وليس بواسطة الكيان.

ويتم إضافة هذه السمات غالبا إلى مخطط كيان العلاقة في أشكال بيضاوية، ولكن من الممكن أن يصبح بهذا صعب الفهم. وإذا قمت بعرضهم في المربع مع الكيان بدلا من ذلك، فلن يصبح التخطيط أسهل في الفهم فقط ولكن يجعل التحويل إلى التصميم الاعتباري أبسط بكثير أيضا. ومع كل السمات في موضع، قد نشاهد التصميم مثل ذلك الموضح في الشكل ٥-٤. من الممكن أن يحتوي الشكل على تخطيط متطور، ولكن هذا المثال كافي لكي يعطي معنى متميز لتصميم قاعدة البيانات.

الشكل ٥-٤

إضافة السمات إلى كيانات قاعدة بيانات Movie.



قد تجد في الغالب ميزات تتطابق مع العلاقات بدل من الكيانات. على سبيل المثال، قد تريد معرفة الممثلون المميزون الذين يعتبروا نجوم في الفيلم، ولكن قد تريد أيضا معرفة الرسم المقرر لهم للظهور فيه أو وقت بدء وانتهاء العمل. يمكن إضافة هذه المعلومات في أشكال بيضاوية كما هو موضح في الشكل ٥-٤.

تلميح لابد أن يكون ظهور السمات للعلاقات دلالة قوية على إمكانية تحويل العلاقة إلى كيان. وعلى الرغم أننا سوف نقوم بتنفيذ هذا الإجراء في القسم التالي، فلابد من تنفيذه هنا أيضا بالمثل.

مراجعة التصميم

والآن يحين الوقت لمراجعة نتائج التخطيطات لكيان العلاقة لمشاهدة كم هو محسوس وسواء يمكن تبسيطه في طريقة ما أو لا. ولتبسيط التصميم، ننظر إلى تخفيض عدد الكيانات لأن جداول أكثر يعني صيانة أكثر ورموز أكثر لتحديثها. ولتخفيض عدد الكيانات لابد من تجميعها.

والأسلوب العام المستخدم هو جمع كيانات تعليمات البرمجة خاصة إذا كانت تحتوي هذه الكيانات على رموز مختصرة وبيان. ويمكن جمع المعلومات ككيان واحد مع حقل جديد لتعريف نوع تعليمات البرمجة وما يمثله. على سبيل المثال، بدل من كيان واحد لكل حالة فيديو وآخر لتصنيف الأفلام، من الممكن إجراء ذلك كله فى كيان واحد كالآتي في الجدول ٥–١.

الجدول ٥–١: توحيد جدولين تعليمات البرمجة في واحد

نوع الرمز	الرمز	البيان
المعدل	PG	توجيه إلى الأصل
المعدل	PG-13	لا توجد أطفال أقل من ١٣
المعدل	NC-17	الشباب فقط
الحالة	RS	محجوز
الحالة	AV	متاح
الحالة	RE	مؤجر

من الممكن الآن أن نأخذ العمودين الأول والثاني كمفتاح. والقرار بوضع فهرس من عدمه على هذا الجدول يأتي في مرحلة التنفيذ، مع أن مع عدد الرموز الصغير هذا، لا يعتبر مهما.

على الرغم أن هذا مثال بسيط، إلا أنه من الممكن أن نطوره أكثر. هل توجد أي جداول أخرى تصف نفس النوع من الكيان؟ ففي هذه الحالة، تقوم جداول الممثلين والمخرجين بوصف كلا الشخصيات وأدوارهم في صناعة الفيلم. ويعتبر جمع هذه الأشياء إلى كيان "مشترك" واحد سهل تماما ويعني فقط أننا بحاجة إلى إضافة سمة تقوم بتعريف دورهم في الفيلم. وقد يكونوا ممثلين أو مخرجين ولكن علينا الآن فتح إمكانات احتواء أدوار أخرى مثل المنتج وكاتب السيناريو بدون الحاجة إلى تكوين جداول جديدة لتناسبهم. وبهذا نكون قد أحدثنا تخفيض في تكاليف الصيانة الإضافية.

التصميم الاعتباري

عندما يكون لديك نموذج مفاهيمي جيد، يمكنك البدء في تحويله إلى تصميم اعتباري. وهذه العملية هي عملية تخطيط الكيانات إلى النموذج الاعتباري والتأكيد على أن العلاقات بين الجداول يمكن تنفيذها كتوليفة من جدولين أو أكثر.

ملاحظة بينما كان أوراكل ٧ قاعدة بيانات علاقية، إلا أنه يشار إلى أوراكل ٨ كقاعدة بيانات كيان علاقية لأنه توجد به القدرة على تخزين هياكل البيانات المعقدة في جداوله. ومع ذلك، فهو ليس قاعدة بيانات موجهة بالوحدة حقيقة. وإذا أدرك المستخدمون طريقة كيان العلاقة، سوف يتم تعديل تخطيط أساليب النماذج لكي تناسبه.

عند ازدياد تعقيد قاعدة البيانات، تصبح فرصة وصول اثنين من المصممين إلى نفس الحل قليلة. ولا يعد هذا ذات أهمية ما دام يتم إتباع خطوط الإرشاد. ومن الظاهر أن ممارسات الأعمال تتغير هذه الأيام بسرعة لدرجة أنك قد تقوم بإجراء تغيرات منتظمة للتصميم في أي شكل على مدار الوقت. والمهم هو أن تأتي بشيء لا يقوم بمعالجة بياناتك بشكل مناسب فحسب ولكن أيضا يسمح بتلك التغييرات الضرورية وكذلك التطور.

وفي حالة كون قاعدة البيانات في مخطط جدول بيانات بسيط، قد تجد أن نفس مجموعة المعلومات يمكن أن تقع في أماكن عديدة مختلفة. وعندما تقوم بتكوين تصميم اعتباري، فأنت تهدف إلى نموذج قاعدة بيانات يحتوي على قليل من تلك البيانات الزائدة. وبهذه الطريقة، لن يكون التصميم الخاص بك خاضع لمشاكل أو شذوذ عند إجراء تغييرات في البيانات. ونموذج العلاقة الذي نستخدمه للتصميم الاعتباري تم تطويره على يد الدكتور E.F.Codd عام ١٩٧٠. ويقوم النموذج الخاص به هذا على جداول ذات بيانات منفصلة بسيطة بعلاقات بين الجداول. وتم تمثيل العلاقات بواسطة مراجع في جدول واحد إلى آخر مع أن هذا يؤدي إلى قدر محدد من تكرار البيانات لأنه لابد من ظهور نفس المفتاح في كلا الجدولين. وعلى مدار سنوات قليلة مضت، أصبح نموذج العلاقة نظام إدارة قاعدة البيانات السائد، على الرغم أنه يتنافس هذا النموذج مع قواعد البيانات الموجهة بالوحدة وكذلك المهجنات مثل أوراكل ٨.

تحتوي مراحل التصميم الاعتباري على المهام الرئيسية التالية:

- التعرف على أنواع السجلات (وخاصة تخطيط الكيانات إلى جداول).
- التعرف على الحقول في تلك السجلات (تعريفات نوع البيانات والأعمدة وهكذا).
- التعرف على أي تبعيات بيانات.
- تنسيق قاعدة البيانات.
- التعرف على المفاتيح.

التعرف على أنوع السجلات والحقول

تقوم طرق كيان العلاقة الموضحة في قسم التصميم المفاهيمي على النموذج العلاقي. ولأننا نطلب أن يكون النموذج الاعتباري نموذج علاقي. فإن التخطيط بين التصميم الاعتباري والتصميم المفاهيمي يكون مباشر. وتعتبر أنواع السجلات مشابهة للكيانات كثيرا بحيث يتمكن معظم المصممين من إجراء تحويل مباشر من واحد إلى آخر. وينطبق نفس الشيء على الحقول التي لابد أن تكون بأية حال نفس سمات الكيان.

التعريف بأي تبعيات بيانات

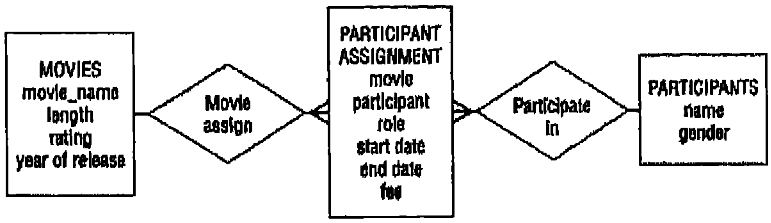
تقوم الكيانات بالتخطيط مباشرة إلى النموذج الاعتباري ولكن لا يتم تخطيط العلاقات، وبدل ذلك، عليك تكوين سمات إضافية بحيث يكون من الممكن تنفيذ العلاقات كمجموعات جداول. تظهر هذه السمات الجديدة غالبا كمراجع من جدول إلى آخر ومن الممكن أن تصبح مفاتيح خارجية في التصميم الحقيقي الخاص بك.

وعليك الانتباه إلى علاقات متعدد لمتعدد لأنهم لا يقوموا بالتنفيذ جيدا في بيئة SQL. ووجود هذه العلاقات علامة على أن العلاقة قد يكون لها بعض الخواص في ذاتها. وعليك اعتبار تمثيل العلاقة ككيان جديد.

على سبيل المثال، في مثال تخزين الفيديو، نجد علاقة متعدد لمتعدد بين الأفلام والمخرجين "يمكن لأكثر من مخرج العمل في الفيلم، بينما يمكن لمخرج العمل في أفلام متعددة". ولكن لا تنسى عمل كيان جديد يسمى Participant_Assignment بعلاقة واحد-لمتعدد لكلا المشترك وجداول الأفلام كما هي موضحة في الشكل ٥-٥. وتعتبر هذه الخطوة هامة على وجه الخصوص إذا كنت تريد تخزين معلومات حول العلاقة مثل أوقات البدء والانتهاء أو المصروفات.

الشكل ٥-٥

تبعيات قاعدة البيانات.



تلميح من الممكن أن يحدث تحويل العلاقات هذا بسهولة فيما بعد في مرحلة التصميم الاعتباري أو في التصميم المفاهيمي. والشيء الهام هو تغطية كل الخطوات حتى إذا لم تواصل النظام المعروض في هذا الفصل.

تطبيع قاعدة البيانات

بمجرد تعريف نسق السجل، يمكنك استخدام أسلوب التطبيع. وهذه هي عملية تبسيط التصميم المعقد إلى مجموعة من العلاقات البسيطة. ويعتبر التطبيع هام لعمل موائمة بين سيناريو حقيقي وشيء ما سوف يناسب قاعدة البيانات العلاقية بشكل مؤثر. توجد قاعدة حسابية شكلية للتطبيع، ولكن ترجع العملية إلى الحس الذي يتناسب مع الآخرين لتحقيق قاعدة البيانات التي:

- تكون سهلة الصيانة.
- من الممكن توسعها بجهد الحد الأدنى.
- لا تحتوي على بيانات تكرارية أكثر من اللازم.
- تقليل حالات عدم توافق البيانات.
- تقليل تأثير التحديث وحذف الشواذ "على سبيل المثال لا تقوم بتحديث كل السجلات المصابة".

يعتبر التطبيع حصيلة العمل الذي تم أساسا بواسطة E.F.Codd والذي تم على أساس نظرية حسابية دقيقة جدا. ولحسن الحظ، لسنا بحاجة إلى فهم أي حسابات لتطبيع قاعدة البيانات. ولقد قام Codd بعرض ثلاثة نماذج قياسية قامت بتعريف خواص التصميم العلاقي الجيد، وهذه هي النماذج القياسية الأولى والثانية والثالثة. قام Codd فيما بعد بالاشتراك مع Boyce في إنتاج النموذج المشترك، Boyce-Codd. والذي يعتبر النموذج الثالث العادي. وفي بحث تالي تم إنتاج النماذج العادية الرابع والخامس. وسوف نناقش بإيجاز هذه الأشياء ولكنها تعتبر بالنسبة للمصممين ذات اهتمام أكاديمي فقط.

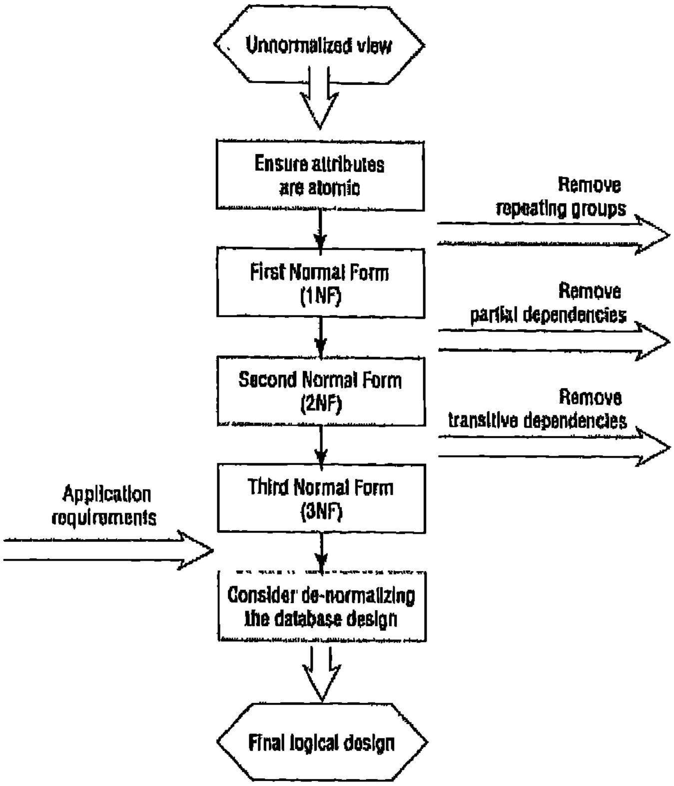
والآن ما الذي نريد تحقيقه مع التطبيع؟ يتم تصميم الأساليب لتأكد من وجود الخواص التالية بقاعدة البيانات:

- وحدة كل السمات.
- عدم وجود حقول متكررة.
- عدم وجود تبعيات مفتاح جزئية.
- إزالة كل التبعيات التي لا يوجد لها مفتاح.

عملية التطبيع موضحة في الشكل ٥-٦ وكل مرحلة سوف يتم وصفها في الأقسام التالية:

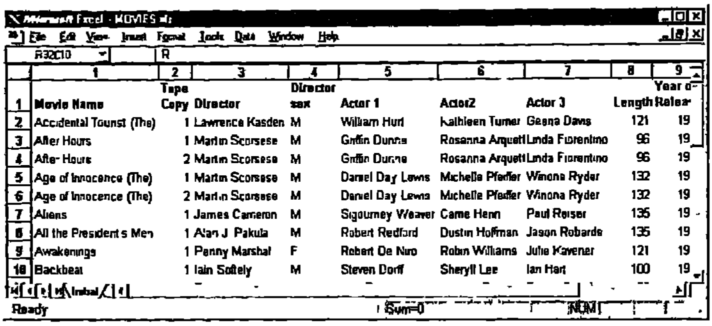
الشكل ٥-٦

عملية التطبيع.



وعن طريق الأمثلة أصبح من الأسهل فهم خواص قاعدة البيانات المرغوبة. وسوف نأخذ مثال مخزن الفيديو الذي يريد أصحابه إعداد قاعدة بيانات صغيرة له لتتبع الشرائط. وبالتطبيق، إذا كنت بصدد إجراء عملية إنتاج أشكال كيان علاقة، لابد أن يكون لديك قاعدة بيانات مطبعة نسبيا لكي تبدأ منها، ولولا هذا المثال لأصبح حدوث أي خطأ محتمل.

ولقد قام صاحب مخزن الفيديو بإعداد البيانات في جداول إكسل الإلكترونية. وتعتبر مايكروسوفت إكسل أكثر قاعدة بيانات مستخدمة بشكل غير رسمي، ومع ذلك فهي تساعد التصميم ضعيف الإمكانات مثل ذلك الموضح في الشكل ٥-٧. ومثال التصميم الضعيف هو عندما تكون نسخة فيديو من فيلم بها نفس المعلومات متكررة مما يجعل عملية الصيانة صعبة "مثل حالة تغيير السعر فأنت بحاجة للبحث عن كل النسخ وتحديثها". مشكلة أخرى هي أن الممثلون يكونوا في ثلاثة أعمدة مما يجعل من الصعب تشغيل الاستعلامات.



	1	2	3	4	5	6	7	8	9
1	Movie Name	Tape Copy	Director	Director sex	Actor 1	Actor2	Actor 3	Length	Year of Relea
2	Accidental Tourist (The)	1	Lawrence Kasden	M	William Hurt	Kathleen Turner	Geena Davis	121	19
3	After Hours	1	Martin Scorsese	M	Griffin Dunne	Rosanna Arquett	Linda Fiorentino	96	19
4	After Hours	2	Martin Scorsese	M	Griffin Dunne	Rosanna Arquett	Linda Fiorentino	96	19
5	Age of Innocence (The)	1	Martin Scorsese	M	Daniel Day Lewis	Michelle Pfeiffer	Winona Ryder	132	19
6	Age of Innocence (The)	2	Martin Scorsese	M	Daniel Day Lewis	Michelle Pfeiffer	Winona Ryder	132	19
7	Aliens	1	James Cameron	M	Sigourney Weaver	Carrie Henn	Paul Reiser	135	19
8	All the President's Men	1	Alan J Pakula	M	Robert Redford	Dustin Hoffman	Jason Robards	135	19
9	Awakenings	1	Penny Marshal	F	Robert De Niro	Robin Williams	Julie Kavener	121	19
10	Backbeat	1	Iain Softely	M	Steven Dorff	Sheryll Lee	Ian Hart	100	19

الشكل ٥-٧ تصميم اعتباري أولي لقاعدة البيانات Movie.

وفي حالة وجود شرائط فيديو قليلة للاعتناء بها، يعمل النظام بكفاءة وتصبح عملية قص ولصق البيانات تعمل بشكل جاهز ومباشر. والخلاصة أن بأكثر ما نضع أفلام في الجداول الإلكترونية بأقرب ما تكون في فوضى أكثر.

التأكد من وحدة السمات

عند القيام بتطبيع تصميم قاعدة البيانات، عليك مراجعة كل سمة للتأكد من وحدتها. ويشير هذا المصطلح "وحدة" إلى السمة التي لا يمكن تقسيمها بدون افتقاد معناها. ومع أن التأكيد على الوحدة ليس وظيفة تطبيع على وجه التقييد إلا أنه جدير بالاعتبار لأنه يعمل لنفس الغرض وهو الحصول في النهاية على تصميم وظيفي مرن. ومثال واحد على تصميم السمات الخاص بك بحيث يكونوا متكاملين هو أسماء المخرجين والممثلين. يوضح تصميم مخزن الفيديو في الشكل ٥-٧ الأسماء الممثلة في حقل واحد والتي تحتوي على الاسم الأول والأخير ويحتمل أن يكون الأوسط للمثل أو المخرج. ومع أنك لست بحاجة للتجزئة، إلا أنه يمكنك تجزئة الأسماء إلى حقلين أو أكثر مثل first_name و last_name والعنوان. وبدون هذه الخطوة سوف يكون من الصعب، على سبيل المثال، فرز last_name.

وبالنسبة للسمات التي تحتوي على أكثر من قطعة للمعلومات فقد تسبب مشاكل، ويوجد اتجاه لتصميم الرموز بحيث يمكنهم تغطية السمات المتعددة. على سبيل المثال، قد يوجد بقسم الموارد البشرية رموز لكل حالة موظف مثل تلك الموضح في الجدول ٥-٢.

الجدول ٥-٢: مثال لجدول الرموز الذي يمكن فهمه بصعوبة

الرمز	البيان
FTT	كل الوقت، مؤقت

الجدول ٥-٢: مثال لجدول الرموز الذي يمكن فهمه بصعوبة

FTR	كل الوقت، منتظم
PTT	بعض الوقت، مؤقت
PTR	بعض الوقت، منتظم
CON	المقاول

يمكنك تصميم هذه الرموز على قاعدة البيانات بسهولة، ولكن كيف يمكن إعطاء قائمة بكل العاملين كل الوقت تحتوي على الأماكن التي يأتي منها المقاولون؟ والمشكلة بهذا التخطيط هي أنه يتم تكوين هذه الرموز بدون التفكير حول إمكانية تناسبهم مع قاعدة البيانات. وبالنسبة للنظام المخطط على ورق، فإن هذه الرموز تعمل جيدا. وتقوم هذه الرموز ببيان ما إذا كان شخص ما كل الوقت أو بعض الوقت ويبين كذلك ما إذا كان العضو منتظم مع المجموعة أم مؤقت. وتتم إضافة تعليمات البرمجة المقاول لتوسيع نظام ضعيف فعلا.

وعليك اتخاذ الحذر عند مثل هذه المشاكل وعليك تجزئة الرموز إلى حقلين أو أكثر "في هذه الحالة "الوقت بالكامل/جزء من الوقت" "المؤقت/المنتظم" وربما حقل آخر "على التعاقد". يعني كل حقل إذا شيء واحد فقط، ويمكنك البحث والفرز بدون مشاكل.

ويعتبر الانتقال إلى السمات الموحدة طريق إلى التسهيل وسوف تتقابل معها في برامج الفيجوال بيسك عندما تريد استخدام واحد من المتغيرات لكي تعبر عن أكثر من شيء. ولابد من التنويه بميزة مخططات ترقيم الرموز التي تحتوي على معني مضمن إلى مدي محدد. على سبيل المثال، من السهل تعديل تطبيق موجود بواسطة متغير يكون دائما موجب بحيث تعني القيمة السالبة شيء ما مختلف قليلا. وقد تقوم بمثال بسيط هو تمرير قيمة رئيسية إلى الإجراء ليتم حذف صف من قاعدة البيانات. وإذا قمت بتمرير قيمة سالبة تعني حذف الصف الرئيسي وكل الصفوف الفرعية المتعلقة به. وتعتبر هذه أسرع طريقة عادة لإجراء هذا التغيير في التطبيق لأنك لست بحاجة إلى إضافة مراقبات معاملة جديدة لأن هذا سوف يسبب الكثير من المشاكل في المستقبل.

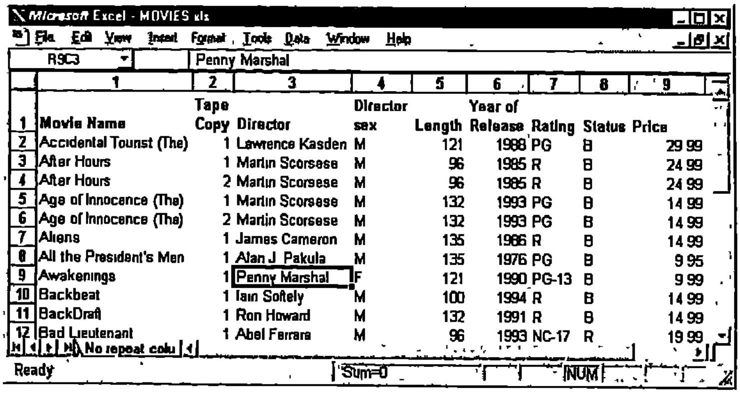
تلميح

ليس من المفضل تجزئة الحقول أكثر من اللازم. ولكن عليك تجزئة الحقول على المستوى الذي يتناسب محـ الاحتياجات. إذا كنت تعلم أنه لن يتم احتواء الاسم الأخير فعليك الاحتواء بهذا الاسم في حقل خاص. ولكن لا يفضل تجزئة العنوان لمجرد أن الشارع ورقم الشارع منفصلين ما لم تكن تقم بتكوين قاعدة بيانات جغرافية.

النموذج الأول العادي First Normal Form 1NF

يتطلب النموذج الأول العادي إزالة أي حقول أو مجموعات متكررة من الحقول. وفي الحقيقة، قواعد البيانات العلاقية "الخالصة" تعوق أي وحدة مع الهيكل الخاص بها. وفي مثال مخزن الفيديو، من الواضح أن لكل فيلم اثنان من الممثلين. فماذا يحدث عندما تريد إضافة ممثل ثالث؟ هل تضيف عمود آخر ثم آخر وآخر؟ وفي وضع إكسل بسيط قد يكون هذا مقبولا ولكن ماذا يحدث إذا أردت البحث عن كل الأفلام المتضمنة نفس البطل المفضل لديك؟ وعليك في هذا الحالة البحث في كل عمود متوقع لكل سجل بينما يصبح الاستعلام الخاص بك أكثر تعقيدا وأقل مناسبة لمولد التقرير الذي قام بإنتاجه. يوجد عامل آخر جدير بالاعتبار وهو المساحة المفقودة عندما يوجد بالفيلم ممثلون أقل من العدد المسموح به في قاعدة البيانات.

وما نحن بحاجة له هو إزالة الأعمدة المتكررة ونقلهم إلى جدول آخر. ولنسمي هذا الجدول "ممثلون". وبالنسبة للوقت الحالي سوف نفترض أن اسم الفيلم يتطابق مع الاسم الفعلي، لذلك استخدم الاسم كأساس لجدول "الممثلون". يوضح الشكل ٥-٨ جدول Movies بدون الأعمدة المتكررة.



Microsoft Excel - MOVIES.xls

R9C3 | Penny Marshal

	1	2	3	4	5	6	7	8	9
1	Movie Name	Tape Copy	Director	Director sex	Length	Year of Release	Rating	Status	Price
2	Accidental Tourist (The)	1	Lawrence Kasden	M	121	1988	PG	B	29.99
3	After Hours	1	Martin Scorsese	M	96	1985	R	B	24.99
4	After Hours	2	Martin Scorsese	M	96	1985	R	B	24.99
5	Age of Innocence (The)	1	Martin Scorsese	M	132	1993	PG	B	14.99
6	Age of Innocence (The)	2	Martin Scorsese	M	132	1993	PG	B	14.99
7	Aliens	1	James Cameron	M	135	1986	R	B	14.99
8	All the President's Men	1	Alan J. Pakula	M	135	1976	PG	B	9.95
9	Awakenings	1	Penny Marshal	F	121	1990	PG-13	B	9.99
10	Backbeat	1	Iain Softely	M	100	1994	R	B	14.99
11	BackDraft	1	Ron Howard	M	132	1991	R	B	14.99
12	Bad Lieutenant	1	Abel Ferrara	M	96	1993	NC-17	R	19.99

No repeat colu

Ready | Sum=0 | NUM

الشكل ٥-٨ تصميم قاعدة البيانات Video Store مع الأعمدة التكرارية التي تمت إزالتها.

وهذا يعتبر في الحقيقة تقييد "وآخرون يعتبرونه نقص" للنظرية العلاقية وSQL بعدم إتاحة الأعمدة المتكررة. فغالبا تكون البيانات تكرارية وعلينا تعديل المعلومات بشكل سطحي إلى نموذج يتناسب مع قاعدة البيانات. وما هو هام إدخال وحدات الفيجوال بيسك إلى الجدول. وقد تحتوي الوحدة على مجموعة أو أكثر. ولوضع مجموعة في جدول، عليك تجزئة الوحدة ووضعها بين جداول منفصلة. ولاستعادة الوحدة، عليك تجزئتها داخل الوحدة المجاورة.

ولقد تم التغلب على هذه المشكلة بواسطة قواعد البيانات الموجهة للوحدة التي تستطيع بسهولة تمثيل مجموعات من البيانات مثلما تفعل فيجول بسيك. ومع ذلك فإن لقواعد البيانات الموجهة

للوحدة مجموعة المشاكل الخاصة بها أيضا. ولقد قدم أوراكل ٨ طريقة علاقية للوحدة تضيف إمكانات موجهة لهيكل قاعدة البيانات العلاقية ولكن بالنسبة للوقت الحالي فليس متاحين لمطور الفيجوال بيسك.

ملاحظة يدعم أوراكل ٨ الآن تكوين الأعمدة التكرارية في أنواع البيانات المتطورة الحديثة: الجداول المضمنة "Nested Tables" وvarray.

إذا لم تقرر تنفيذ تصميم قاعدة البيانات مع الأعمدة المتكررة، سوف تكون قادر على الأقل على التعامل مع الأعمدة في الفيجوال بيسك. وعلى الرغم أن SQL، قبل أوراكل ٨، ليس بها معالجة مجموعة من السمات إذا تمت قراءة البيانات إلى مجموعة تسجيل الفيجوال بيسك، فإنه يمكن الوصول إلى الحقول بواسطة الفهرس. وسوف نناقش إمكانية القيام بهذا في الفصل ١٤ "وحدات الوصول إلى البيانات".

وإذا أردت القيام بتمثيل مصفوفة في جدول، فمن الأفضل عمود واحد لكل اتجاه في المصفوفة وعمود واحد أو أكثر لقيم البيانات "راجع الشكل ٥-٩". ولاسترداد البيانات، عليك التحديد بواسطة مجموعة السكريبت الفرعية. وعلى الرغم أن هذه الطريقة سوف تعمل، فلابد من البحث عن بديل لأن الجدول لا يعتبر التصميم أو التنفيذ الأمثل.

الشكل ٥-٩ استخدام جدول لتصميم مجموعة.

X	Y	Value
1	1	25
1	2	15
2	2	0

Array(1,1) = 25
Array(1,2) = 15

الاهتمام النهائي بخصوص إزالة الأعمدة المتكررة هو كيفية إعداد تقرير بعناوين البيانات المنتشرة بالصفحة. على سبيل المثال، قد نريد نماذج مبيعات شهرية حسب الأشهر الموجودة بالصفحة، فإن أوراكل، لسوء الحظ، ليس به وظيفة جدولة متقاطعة، لذلك أفضل خيارات هي وضع البيانات في تنسيق مصفوفة في تعليمات البرمجة VB أو وضعها في إجراء مخزن أو استخدام مولد تقارير يمكنه إجراء ذلك.

تكرار الأعمدة بصورة فعلية

في كتب النصوص على إعداد نماذج قواعد البيانات، أكثر مثال موضح تكرار الأعمدة هو Invoices وLine_Items في نظام الفواتير. وفي هذه الحالة، تعتبر تجزئة Line_Items من Invoices "علاقة أساسي تفصيل" هى أفضل شيء تقوم به. وسوف تعرف عدد Line_Items الموجودة لكل Invoice والتي تقوم بإفساح المجال لعدد كبير من المساحات المفقودة في معظم المواقف. وأنكر أيضا أن قواعد البيانات العلاقية لا يوجد لها دعم لمعالجة الجداول التكرارية حتى إذا قمت باحتوائهم.

ولكن يوجد حالات قد تكون بحاجة لاعتبارها. فلا يوجد شيء خطأ معروف بالنسبة للأعمدة التكرارية كمفهوم، وكل ما في الأمر أن النموذج العلاقي لا يمكنه معالجة هذه الأعمدة جيدا. ولقد تواجهت مع موقف لمرات قليلة وهو الحاجة إلى تكوين قاعدة بيانات لاحتجاز البيانات في كل ساعة مع قيمة مختلفة لكل ساعة. فهل نقوم بتطبيع قاعدة البيانات بحيث يكون هناك صف لكل ساعة، كما هو موضح أدناه، أو أن نقوم بوضع ساعات اليوم على صف واحد كما هو موضح أدناه أيضا؟

Date	Hour	Value
12/01/98	1	101
12/01/98	2	105
12/01/98	3	116

البديل الآخر هو استخدام قاعدة البيانات التوجيه الوحدة التي تسمح بالبيانات الهيكلية. وفي مثل هذه الحالات، لا يكون لبعض من سمات البيانات متصل وسوف نستخدم الآخرين في كيان الترابط.

Date	Hour1	Hour2	Hour3	Hour4	etc.
12/01/98	101	105	116	116	
12/02/98	105	106	108	120	
12/02/98	206	206	205	NULL	

وكان العمل التمهيدي لقاعدة بيانات تم تطبيعها وقد سمحت بتخزين معلومات أخرى لكل ساعة مثل اسم المستخدم وختم الوقت لآخر تحديث والإشارة إلى جدول التعليقات. فمن ناحية التصميم، تعتبر طريقة التطبيع أفضل طريقة تتبع

[illegible]

Date	Start	Stop	Value
12/01/98	0	1	101
12/01/98	1	2	105
12/01/98	2	6	116

[illegible]

ملاحظة

وباستخدام النموذج الأول العادي، تم الانتقال من جدول واحد "الفيديوهـــات" إلى جدولين "الفيديوهات والممثلين".

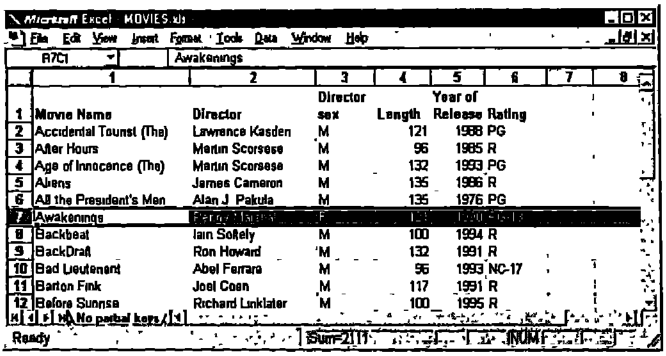
النموذج الثاني العادي 2NF Second Normal Form

ولفهم النموذج الثاني العادي، لابد من فهم المفاتيح أولاً. والمفتاح هو عبارة عن حقـــل أو أكـــثر يمكن استخدامه "أو استخدامها" لتعريف السجل "وليس من الضروري أن يكونوا فريدين، وســوف

نناقش المفاتيح المختارة المميزة فيما بعد".عند تجميع أكثر من حقل لتكوين مفتاح، فـــهذا يســـمى المفتاح المركب. وليس من الضروري أن تكون هذه الحقول مجاورة لبعضها.

وفي مثال مخزن الفيديو، أكثر مفتاح واضح للفيديوهات هو اسم الفيلم المرتبط برقم النســخة. ومع هذين الحقلين، من الممكن تحديد أي فيديو في المخزن بالكامل. وسوف تلاحظ أن بعض من الحقول التي لا ترتبط بالمفاتيح مثل سنة أو مدة الإصدار وهذا يعتمد فقط على الفيلم ذاته. وعندما تعتمد السمات على جزء واحد فقط من المفتاح، فهذا يسمى التبعية الجزئية للمفتاح.

يقول النموذج الثاني العادي بعدم تفضيل وجود تبعيات جزئية للمفتاح. وفـــي مثـــال مخـــزن الفيديو، يعتمد تاريخ الإصدار والمدة على جزء فقط من المفتاح بالكامل: اسم الملف. ولكي نكيف أنفسنا مع النموذج الثاني العادي، لابد من نقل هذه الحقول إلى جدول آخر، الذي ســـوف نســـميه "Movies" "راجع الشكل ٥-١٠". وجدول الممثلين موجود فعلا في هذا النموذج لأنه تم تكوينـــه بهذه الطريقة.



	1	2	3	4	5	6
1	Movie Name	Director	Director sex	Length	Year of Release	Rating
2	Accidental Tourist (The)	Lawrence Kasden	M	121	1988	PG
3	After Hours	Martin Scorsese	M	96	1985	R
4	Age of Innocence (The)	Martin Scorsese	M	132	1993	PG
5	Aliens	James Cameron	M	135	1986	R
6	All the President's Men	Alan J. Pakula	M	135	1976	PG
7	Awakenings	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
8	Backbeat	Iain Softely	M	100	1994	R
9	BackDraft	Ron Howard	M	132	1991	R
10	Bad Lieutenant	Abel Ferrara	M	96	1993	NC-17
11	Barton Fink	Joel Coen	M	117	1991	R
12	Before Sunrise	Richard Linklater	M	100	1995	R

الشكل ٥-١٠ إزالة تبعيات المفتاح الجزئية لإنتاج النموذج الثاني العادي.

[illegible]

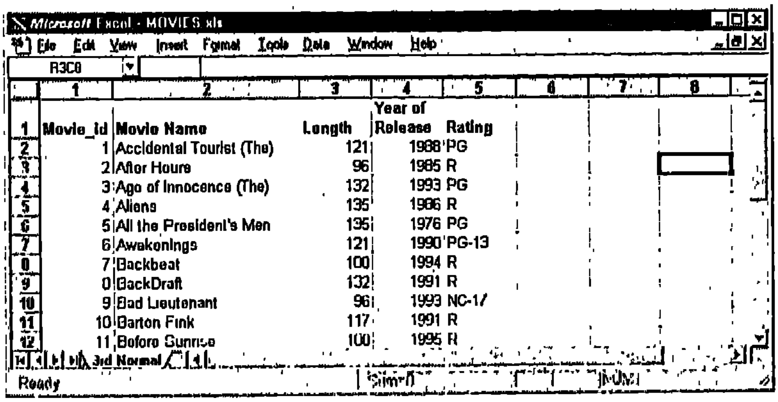
والمشكلة الثانية لتخزين سجل تاريخي للتغيرات أكثر تعقيداً لأنه لابد من إعداد جدول جديد للاحتفاظ بالسجلات التاريخية، وسوف نعرف كيف يمكن الحصول على قيمة لتاريخ محدد من هذا الجدول في الفصل ٧، "الاستعلامات" وهذه الخطوة ليست الأبسط في الاستعلامات، لذلك تخزين السعر الحالي في جدول آخر عادة يعطي الأداء مكسب.

ملاحظة باستخدام النموذج الثاني العادي، انتقلنا من الجدول الثاني إلى الجدول الثالث: الفيديوهات والممثلون والأفلام.

النموذج الثالث العادي 3NF Third Normal Form

النموذج النهائي المستخدم هو النموذج الثالث العادي، ومن الممكن أن تجد بعض الصعوبة في فهمه. فمن الناحية الفنية، يتم إزالة كل التبعيات الانتقالية. ولكي نعرضه في صورة مبسطة، لابد من تجنب حالة تغيير حقل غير مكون بواسطة مفتاح الذي يعني ضرورة تغيير حقل غير مكون بواسطة مفتاح آخر. الشيء الآخر هو النظر إلى النموذج الثالث العادي، من حيث أن كل حقل غير مكون بواسطة مفتاح لابد أن يعتمد كلية على المفتاح الأساسي ككل ولا شيء آخر.

وفي المثال الذي بين أيدينا، يوجد نوع المخرج في نفس السجل مثل المخرج. ويعتمد عمود النوع على حقل المخرج وليس على حقل الأفلام لذلك فهو غير متضمن في الجدول. وبالنسبة لتصميم النموذج الثالث العادي، علينا أن نأخذ الحقول التي تصف المخرج خارج الجدول ثم نقوم بوضعهم في جدول المشتركين مع بيان الدور الخاص بهم "راجع الشكل ٥-١١".



Microsoft Excel - MOVIES.xls

Movie_id	Movie Name	Length	Year of Release	Rating
1	Accidental Tourist (The)	121	1988	PG
2	After Hours	96	1985	R
3	Age of Innocence (The)	132	1993	PG
4	Aliens	135	1986	R
5	All the President's Men	135	1976	PG
6	Awakenings	121	1990	PG-13
7	Backbeat	100	1994	R
8	BackDraft	132	1991	R
9	Bad Lieutenant	96	1993	NC-17
10	Barton Fink	117	1991	R
11	Before Sunrise	100	1995	R

3rd Normal

الشكل ٥-١١ إزالة التبعيات الانتقالية مثل حقل Director الذي يؤدي إلى النموذج الثاني العادي.

ملاحظة باستخدام النموذج الثالث العادي، انتقلنا من ثلاثة إلى أربعة جداول: الفيديوهات والممثلين والأفلام والمخرجين.

تخزين البيانات المشتقة

إذا قمت بتخزين أي قيم في قاعدة البيانات يتم حسابها أو بطريقة أخرى اشتقاقها من قيمة أخرى، إذا لن يتفق التصميم مع النموذج الثالث العادي. ومع ذلك فإن الاحتياجات تكون ملحة لإجراء ذلك على وجه التحديد. وتحدث هذه الحالة بشكل عادي عند استخدام موارد خادم كثيرة جدا لحساب قيمة بشكل متكرر، وكذلك سوف تحدث أينما تصبح القيمة المشتقة غير قادرة على إعادة التكوين في وقت بعد ذلك.

ومن ناحية الأداء، أي علاقة أساسي تفصيل سوف يكون منتخب لتخزين قيمة مشتقة. وفي قاعدة البيانات Movie حيث توجد علاقة واحد لمتعدد بين الأفلام وكيانات الفيديو، قد يصبح من الممكن تخزين العدد الإجمالي للفيديوهات في كيان الأفلام، وهذا شيء لابد أن يكون مدروس جيدا في مرحلة التصميم. وعلى الرغم من إمكانية إضافته في وقت متأخر، يعتبر هذا النوع من القيمة المشتقة بحاجة لصيانته بحيث تعتبر التفاصيل التي تقوم بتكوينه في اتفاق مع قيمة الإجمال.

وفي قاعدة بيانات تخزين الفيديو، لن يكون هناك إقبال كبير على معرفة الرقم الإجمالي لنسخ الفيديو لفيلم، ولكن في حالات أخرى، قد يصبح هذا القرار مصيري بالنسبة للأداء، والتحسين الحقيقي الذي تحصل عليه من الصعب تعيينه لأنه يعتمد على عدد مرات تحديث القيمة المشتقة وعدد مرات قراءته وعدد السجلات التي بحاجة للقراءة لحساب القيمة.

أساليب التطبيع الأعلى

تقوم النماذج الثلاثة العادية الأولى بإزالة معظم الزيادات من التصميم. وكان البحث الذي كان في ١٩٨٠ قد أدى إلى أساليب أخرى متعددة وتتضمن نموذج Boyce-Codd العادي "نموذج مدعم من النموذج الثالث العادي الذي يقوم بإزالة كل التبعيات الانتقالية" والنموذج الرابع "الذي يحذف تبعيات القيمة المتعددة في الجداول" والنموذج الخامس العادي. إذا قمت بإرسال معظم قواعد

بيانات التصميم العادية، فعليك البحث عن الأساليب، ومع ذلك فإن الحالات التي تريد فيها استخدام النماذج العادية الأعلى تصبح أقل شيوعا تدريجيا.

تبسيط التصميم

والآن انتهينا من مرحلة التطبيع، وعلينا مراجعة التصميم الآن لنرى ما إذا كان من الممكن تبسيطه أو تحسين مرونته. ففي الجزء الأول من مرحلة التصميم الاعتباري وعند تحديد أي تبعيات بيانات، فقد لاحظنا أن جدولا الممثلون والمخرجون متشابهان جدا في المحتوى. وقد قمنا بتكوين جداول جديدة تسمى Participants للأفراد وParticipant_Assignment للأدوار القائمين بها في الأفلام. وعلى الرغم من وجود نفس العدد من الجداول، فقد تم تحسين مرونته جديا من حيث إمكانية احتوائه معلومات أكثر عن التخصيصات وإمكانية احتواء أدوار مختلفة على الفيلم.

ملاحظة وهنا سوف نضيف في جدول Customer، الذي لم يكن جزء من الجدول الإلكتروني الأصلي. والآن قد انتهينا من خمسة جداول في التصميم: الفيديوهات والأفلام والمشتركين وParticipant_Assignment والعميل

إلغاء تطبيع قاعدة البيانات

باتخاذ تصميم قاعدة البيانات إلى النموذج الثالث العادي، فقد انتقلنا من جدول إلى خمسة جداول. ولقد تعمقنا للحصول على تصميم خالص ولكن كان هذا على حساب الأداء. فمن الناحية العقلية، عليك الإشارة إلى أكثر من ثلاثة أو أربعة جداول عند تحديد البيانات في أي وقت من قاعدة البيانات. وأي زيادة تكون دليل على أن الأداء به عيب وتصبح الاستعلامات أكثر تعقيدا.

والحل لمشكلة الأداء هو عدم إجراء خطوة تطبيع أو أكثر أو إلغاء تطبيع قاعدة البيانات. ولابد من إجراء هذا بعناية بالطبع. والمنطقة التي قد تريد النظر إليها هي جداول الرموز وخاصة إذا كنت تقوم باستخدام العديد منهم. وقد يعتبر استخدام الاسم بالكامل بدل من تعليمات البرمجة الجدير بمصروفات فقد مساحة التخزين.

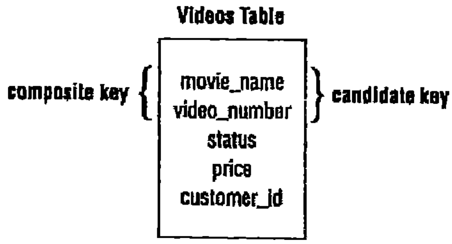
وإمكانية أخرى للأداء الذي تم تحسينه هي استخدام الأساليب التي تستخدم لتطوير تخزين البيانات على الرغم أن الفكرة موجودة من سنين. يمكنك إنتاج جدول جديد أو قاعدة بيانات منفصلة باستخدام كل شيء تم إلغاء تطبيعه. وتفضل طريقة إلغاء التطبيع هذه لنظم دعم القرار فقط، وتم إجراء التحديثات فقط لجداول القاعدة، وآنذاك يمكن إعادة تحميل قاعدة البيانات التي تم إلغائها كل ليلة بواسطة البيانات المشتقة من جداول القاعدة التي تم تحديثها. وتعتبر هذه الطريقة

جيدة غالبا بالنسبة للتقارير. ويمكنك كذلك صيانة جداول البيانات المشتقة في وقت معين، على الرغم من ضرورة التأكد من صيانة توافق البيانات.

ومسألة إلغاء تطبيع قاعدة البيانات ليس بالشيء الذي يؤخذ على ظاهره، فلابد أن تكون معد لدعم القرار الخاص بك لتكوين قاعدة البيانات التي لا توجد حتى في النموذج الأول العادي. ولابد أن توثق النموذج العادي الثالث وتضيف مبررات لإلغاء تطبيعه. والأسباب الرئيسية سوف تكون المرونة والأداء ولكن يفضل أن تحصل على بعض التثبيت من الاختبارات التي قمت بها.

استخدام المفاتيح

والآن تعتبر قاعدة البيانات في نموذج التطبيع "وفي خمس مرات مثل العديد من الجداول"، وعلينا أن ننظر غلى كيفية تمييز كل صف من الجدول. فلابد أن يكون كل صف في الجدول العلاقي مختلف ولابد من وجود سمة واحدة على الأقل تتطابق مع الصف بشكل منفرد. وتسمى هذه السمات المفاتيح المرشحة. ومن حين لآخر، لابد من جمع أكثر من سمة، ويسمى هذا "المفتاح المركب" أنظر الشكل ٥-١٢.



الشكل ٥-١٢

يوضح تصميم الجدول المفاتيح المركبة والمرشحة.

وفي هذه المرحلة من تصميم تخزين الفيديو، لابد من البحث عن كل المفاتيح المرشحة للجداول. وبالنسبة لجدول Participants، قد يكون هذا اسم الممثل أو المخرج، على الرغم أنه سوف تتم تغطية الأسباب لعدم استخدام اسم في القسم التالي. وفي حالة جدول Videos، علينا تقرير ما إذا كنا سوف نستخدم movie_name ورقم الفيديو. والبديل الأفضل هو استخدام مفتاح مميز للفيديو وآخر للفلم.

المفاتيح الأساسية

عند مراجعة كل المفاتيح المرشحة، نريد اختيار أكثر واحد مناسب للمفتاح الأساسي. ويقوم المفتاح الأساسي لكل جدول بتحديد كل صف في الجدول بشكل مميز ولا يمكن أن يكون خالي "بمعنى لابد أن تكون له قيمة". من الممكن أن يكون بجداول أوراكل مفتاح أساسي واحد محدد

ولكن يمكننا تحديد مفاتيح فريدة للمفاتيح المرشحة الأخرى. من الممكن أن تكون هـذه المفـاتيح الفريدة خالية، ولكن أثناء التطبيق سوف تقوم غالبا بتحديدهم بحيث لا يكونوا خالين.

تلميح يفضل تصميم وتنفيذ المفتاح الأساسي لكل جدول. ومع أنها قد لا تستـخدم دائما في الاستعلامات، تستطيع الفيجوال بيسك تكوين مجموعات سـجلات قابلة للتحديث مقابل الجداول مع المفاتيح الأساسية.

المفاتيح البديلة

ماذا يحدث إذا لم تتمكن من الحصول على المفتاح الفريد المناسب الذي يمكن أن يكون المفتـاح الأساسي؟ وهذا يحدث كثيرا وخصوصا عندما تريد إجراء تغيير في قيمة المفتاح الأساسي. علـى سبيل المثال، ماذا يحدث إذا كان لدينا إعداد قاعدة بيانات يقوم فيه الاسمان الأول والأخير بتكوين المفتاح الأساسي، وتنتهي بتلك القيمة المستخدمة في كل جدول يشير إلى الممثل. وإذا تغير اسـم الممثل، ماذا نفعل؟

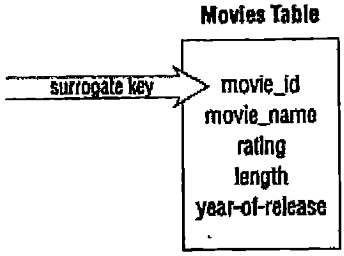
والمشكلة الثانية هي مشكلة التفريد. وبطبع قد لاحظت هذه المشكلة فعلا: كون اسـم الفيلـم متناسب مع المفتاح طالما هو فردي. من الممكن جمع الاسم مع السنة، ولكن في هذه الحالـة، لا يعتبر بالضرورة كون الاسم فريد. ولجعل المسائل أكثر صعوبة، ماذا يحدث في حالة وجود نفس الاسم لأكثر من ممثل؟

تحذير تعتبر الأسماء عادة اختيار ضعيف بالنسبة للمفتاح الأساسي بسبب إمكانيـة التغيير وصعوبة التأكد من كونهم فريدين وسوف يكونوا كذلك.

وطريقة واحدة يمكن استخدامها لمعالجة هذه المسائل وهي تخصيص مفاتيح بديلة. والمفتـاح البديل هو مفتاح فريد نقوم بتكوينه ويكون مستقل عن البيانات "راجع الشـكل ٥-١٣". وسـوف نقوم باستخدامه بشكل عادي كمفتاح أساسي. وتستخدم هذه الطريقة دوما في مايكروسوفت أكسس وكذلك في نماذج مختلفة متاحة في أوراكل: نستخدم مولدين مسلسـلة لتوفـير القيمـة للمفتـاح الأساسي. وبرغم من وجود مفاتيح أساسية بشكل كافي تقوم بتغيير القيمة إلا أنه يتـم اسـتخدام المسلسلات بكثرة. وسوف نتعامل معهم في الفصل التالي لأنهم يعتبروا إصدار تنفيذ فعلي.

الشكل ٥-١٣

تحديد المفتاح البديل في هذه الحالة movie_id.



ولكن عليك إدراك وجود بعض العيوب القليلة في المفاتيح البديلة ومنها صعوبة وضع سجل في حالة عدم معرفة المفتاح" الذي لا يمكنك معرفته عن طريق البيانات مما يتطلب منك البحث في كل الجدول" وفي هذه الحالة عليك وضع فهرس غير فريد على تلك السمة. على سبيل المثال، إذا كنا نستخدم المفاتيح البديلة على جدول Participants ويستخدم هذا الجدول كمفتاح أساسي، فقد تريد أيضا وضع فهرس على سمة movie_name لكي يمكنك البحث بسرعة عن المعلومات الشخصية. وسوف تتم مناقشة هذه الفهارس في الفصل التالي.

المشكلة النهائية بالنسبة للمفاتيح البديلة وهي عندما تريد دمج بيانات من مصدرين مختلفين "قد تكون تنسخ بيانات إلى مستودع بيانات أو قد تكون الشركة تم دمجها مع أخرى". وفي حالة كون جدولان متساويان يستخدمان مفاتيح بديلة، فقد يحدث تكرار المفتاح.

ملاحظة

تعتبر المفاتيح البديلة وسيلة مستخدمة بشكل شائع على الرغم أن العديد من المصممين ليس لديهم أي خلفية عنهم من قبل. على سبيل المثال، يشجع مايكروسوفت أكسس على استخدامهم عن طريق جعل الحقل المقابل متاح. وهذا النوع من الحقول أسهل في الاستخدام من تكوين المفتاح وإجراء فهرس له.

وعلى الرغم من المشاكل القليلة التي قد تسببها المفاتيح البديلة، إلا أنه توجد المميزات الكافية للتوصية بهم في معظم المواقف. وعندما ننتقل إلى تنفيذ قاعدة البيانات في الفصل التالي، سوف نقوم بتكوين مفاتيح بديلة لكل الجداول. وسوف يتم استخدام هذه المفاتيح كمراجع لمفاتيح الربط في أي جداول متعلقة.

مفاتيح الربط

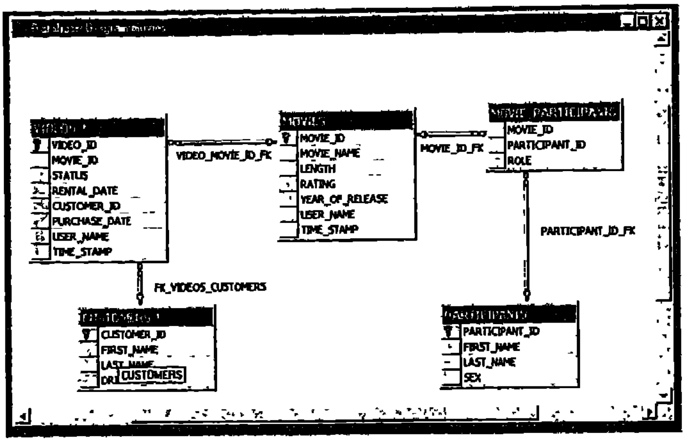
مفتاح الربط هو مرجع من جدول ابن أو جدول تفصيل "مثل جدول Videos" يرجع إلى صف في جدول الأب أو الجدول الأساسي "مثل جدولMovies". ويتم تحديد حالة الجدول بكونها أب أو ابن بناءا على أي جانب من جوانب علاقة واحد- لمتعدد يوجد عليها هذا الجدول. والأب هو "واحد" والابن هو "متعدد". ويقوم الابن بتحديد الأب باحتوائه على مفتاح الربط الذي يعتبر سمة تربط الصف الموجود في الابن مع السمة المساوية في جدول الأب الذي لابد أن يكون مفتاح أساسي. على سبيل المثال، لابد أن يحتوي جدول Videos على مفتاح ربط يحتفظ بقيمة مساوية للمفتاح الأساسي لجدول Videos. ومن الغريب أن تفكر فيه بهذه الطريقة، ولكن وحدة الأب ليس لديها أي فكرة عن المرشحين الذين سيكونو أبناء لها، وعلى الأبناء أن يكونوا معروفين كذلك.

ولكن يوجد قيد خاص لأوراكل لمراجعة صلاحية هذا الترشيح. إذا كان الأب المرشح غير موجود، فإن قيد مفتاح الربط "ويسمى أيضا قيد التكامل المرجعي" لن يسمح بحفظ هذا السجل. ومن الناحية الأخرى، إذا أردت حذف أب من ابن موجود، فإن نفس القيد لن يسمح لك بذلك. وسوف نقوم بتغطية مفاتيح الربط بتفاصيل أكثر في الفصل التالي.

تشغيل عملية التصميم آليا

إلى هذه النقطة قمنا بالتعرض للتصميم بكونه عملية يدوية. ويمكنك استخدام أي من أدوات الشاشة الرئيسية "سطح المكتب" حسب اختيارك مثل أدوات الرسم والجداول الإلكترونية. واعتقد أن هذه الطريقة تعتبر الأفضل لتعلم الأساليب. ومع ذلك، إذا قمت بإجراء عمل تصميم، فقد تكون بحاجة للاحتفاظ بأداة وخصوصا إذا كان تصميمها يتناسب مع اختياراتك. وتسمى هذه الأدوات عادة أدوات CASE "هندسة أنظمة الكمبيوتر". بالإضافة إلى أننا نتحدث عن الاستثمار الهام في البرامج.

بالنسبة للمستخدمين الذين قاموا بشراء طبعة Enterprise من فيجوال بيسك ٦، يمكنهم الوصول إلى Database Designer وDatabase Diagrams. وسوف ننظر إلى هذا المنتج في القسم التالي. يوضح الشكل ٥-١٤ مثال لقاعدة البيانات التي يمكنك الحصول عليها باستخدام Database Designer.



الشكل ٥-١١ العلاقات في قاعدة البيانات video_store كما هو واضح من مخطط بيانات VB6's.

خلاصة

لقد قمنا في هذا الفصل بتغطية الخطوتين الأولى والثانية من تصميم قاعدة البيانات. التصميم الاعتباري والتصميم المفاهيمي. ويقوما هذان التصميمان بتكوين نواة نموذج البيانات. وتوجد طرق مختلفة لإجراء هذه الخطوات، وقمنا بالتعامل مع إعداد النموذج العلاقي من خلال ذلك ونتيجة ذلك، يوجد بعض التداخل بين المراحل الاعتبارية والمفاهيمية. وبالتطبيق، تساعد كل مرحلة الأخرى، ولا تقوم في النهاية بالحصول على مجهود مكرر، ولكن تقوم بالحصول على تصميم مستحدث من جديد.

وسواء كان البدء من الجداول الإلكترونية أو أداة CASE النهائية مثل Oracle Designer ٢٠٠٠، فإن النقطة الهامة هي التأكد من أنك تقوم بتنفيذ التصميم. والاستثمار الذي سوف تحققه في وقت مراحل التصميم سوف يكون أكثر من التصميم الذي تحققه بعد التنفيذ.

٦

تنفيذ قاعدة البيانات

وبهذا تكون قد أتممت عمل التصميم الذي بدأت فيه من الفصل السابق. والآن أنت جاهز للانتقال إلى تنفيذ التصميم الفعلي لقاعدة البيانات. وقد تغتري بالانتقال قدماً إلى هذه المرحلة بدون التوقف لوهلة عند المرحلتين السابقتين: التصميم الاعتباري والنصميم المفاهيمي. ولهذا فإذا قمت بتجاهل هاتين الخطوتين، فسوف يعود عليك هذا بالمشاكل فيما بعد.

تعتبر الميزة الرئيسية لتجهيز قاعدة العمل قبل الوصول إلى هذه المرحلة هي الاطمئنان أنه يتم تصميم قاعدة بيانات ذو كفاءة وليس مجرد تجميع عشوائي للأشياء. بداية، سوف تبين قاعدة البيانات هذه للعميل أنك قمت بإجراء مهمة موثوق بها. ولذلك فإن الهدف الرئيسي هو أنه بمجرد تصميم قاعدة البيانات بشكل جيد، فإن التصميم الحقيقي سوف يكون بالفعل أفضل وسوف يكون التنفيذ أفضل وسوف تقوم بتوفيق احتياجات الأعمال الفعلية بدقة أكبر وسوف تكون على المدى البعيد أرخص وأسهل في عملية الصيانة.

وفي الفصول السابقة، شاهدنا المفاهيم التي توضح الأشياء التي تسهم في تكوين قاعدة بيانات أوراكل وكذلك كيف يمكن تصميم قاعدة بيانات. وفي هذا الفصل، سوف نقوم بتنفيذ قاعدة بيانات محزن الفيديو التي قمنا بإجراء تصميم مفاهيمي واعتباري بالفعل لها. ومع أن التصميم بسيط جدا، إلا أن الطرق تنطبق على كل قاعدة بيانات تقوم بالتعامل معها.

ويوجد أكثر مما تتخيل لاستخدامه في التنفيذ الحقيقي لقاعدة البيانات لأوراكل، خاصة إذا كان المصدر خلفية على أساس الملف، مثل مايكروسوفت أكسس وفوكس بروا. وليس من الكافي فهم كيفية إضافة أنواع الملفات إلى جدول بنقرة زر: وإذا لم تتبع سياسة الانتقال، سوف تصل في النهاية إلى تنفيذ قاعدة بيانات ضعيف يكون بحاجة مستمرة للصيانة.

وعلى كل، سوف يتم معالجة هذه المهام بواسطة مسئول قاعدة البيانات يعمل معك دوما، وخصوصا فى تنظيم أكبر حيث يتم التحكم في كل شئ عن قرب. وكلما تزداد خبرتك، كلما ازدادت حريتك في التحكم في هذه الأشياء بنفسك. وهاك بعض الخطوات العامة التي يجب عليك التفكر فيها عند تنفيذ قاعدة البيانات.

- تكوين جداول الملفات لقاعدة البيانات. وتعتبر تلك الجداول الهياكل الاعتبارية الأولية التي يجب أن تقوم بإعداد وتخصيص مساحة القرص الحقيقية لها للاحتفاظ بالمتبقي من هياكل قاعدة البيانات.
- تحديد أي قيود يستدعيها التصميم الخاص بك.
- تحديد أي فهارس تحتاجها لأغراض الأداء.
- تكوين أي مسلسلات تحتاجها لتزويد قيم فريدة لأعمدة المفاتيح.
- إضافة أعمدة محددة إلى الجدول الخاص بك لتعيين أغراض التدقيق.
- مراجعة وحدات نخطط قاعدة بيانات أخرى التي قد تريد تنفيذها.

- تكوين أي مستخدمين هم بحاجة إلى الوصول إلى البيانات الخاصة بك، ومنح أي امتيازات ضرورية لهم.
- استخدام مصمم قاعدة بيانات مايكروسوفت.
- والآن علينا تطبيق كل واحدة من هذه الخطوات.

تكوين ملفات الجداول

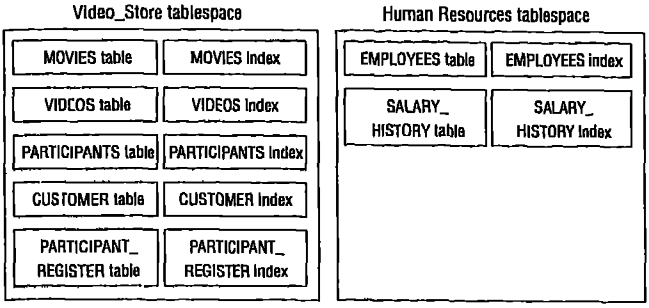
قدمنا في الفصل الثاني مفهوم جدول البيانات، أكبر هيكل بيانات اعتباري في قاعدة بيانات أوراكل. تقوم ملفات الجداول بتخزين وحدات قاعدة البيانات وتحتوي على ملف حقيقي واحد على الأقل لكل ملف جدول. يمكنك إضافة ملفات أكثر أو أن تجعل الملفات تزداد حسب الطلب. تحتوي ملفات الجداول على مقطع أو أكثر، كل واحد منها يحتوي على جدول أو فهرس أو عنقود أو مقطع إلغاء تعديلات أو مقطع مؤقت.

تستخدم ملفات الجداول لفصل طبقات مختلفة من البيانات، وقد تكون بحاجة لإعداد ملف جداول أو أكثر لكل تطبيق يستخدم قاعدة البيانات "راجع الشكل ٦-١". ويجب أن يقوم تصميم قاعدة البيانات بتحديد أنواع أو طبقات البيانات التي سوف يتم إدخالها إلى قاعدة البيانات واختيار أفضل مكان لكل نوع. وإذا كانت هذه هي المرة الأولى لاستخدامك لقاعدة بيانات أوراكل وتعد مرحلة إرشادية لما يليها، فقد تكون بحاجة لإعداد جدولين ملفات للتطبيق: واحد للبيانات وآخر للفهارس. وعندما تقوم بإضافة جداول جديدة، تأكد أنه إذا كانت هذه الجداول تنتمي إلى تطبيق منفصل، فتوضع هذه الجداول في ملف جداول منفصل.

تتضمن مميزات استخدام ملفات الجداول المنفصلة لفصل البيانات القدرة على وضع البيانات على محركات قرص منفصلة لأسباب ترجع إلى الأداء. ولكن دائماً ما يكون إغفال إمكانية عزل ملف جداول منفصل مع إمكانية تشغيل باقي قاعدة البيانات. على سبيل المثال، في حالة فشل القرص المتضمن ملف جداول محاسبي، فإنه يمكن تشغيل قاعدة البيانات بدون ملف الجداول هذا. حاول التفكير حول هذه الأشياء مقدماً. فمن الأصعب جداً تغيير قاعدة البيانات في حالة الإنتاج وملئها بالبيانات بسرعة.

شكل ٦-١

استخدام ملفات الجداول لتخزين وحدات قاعدة البيانات



ملاحظة يتم التحكم في الوصول إلى ملفات الجداول، مثل أي شئ في أوراكل، حسب امتيازات وحصص نسبية. وسوف نناقش هذا الموضوع في هذا الفصل بقسم "إعداد الامتيازات".

إذا كان لديك معرف مستخدم جديد، قد لا يمكنك الوصول إلى أي من ملفات الجداول، أو للاحتمال الغالب، سوف يتم احتجازهم لتطبيقات محددة. للتجربة في الأمثلة في هذا الفصل، فقد تحتاج إلى الوصول إلى من ملف جداول واحد على الأقل، ولكن عليك الانتباه إلى عدم تكوين أي وحدات لقاعدة البيانات في ملف الجداول SYSTEM. فلن يسمح لك محلل قاعدة بيانات خبير بتكوين بيانات في هذا المكان بأي طريقة، لذلك سوف تكون بحاجة إلى الحصول على ملف الجداول الخاص بك.

ولتكوين ملف الجداول الخاص بك، يجب عليك استخدام جملة تكوين ملف الجداول التالية:

```
CREATE TABLESPACE video_store
            DATAFILE 'c:\database\test.db' SIZE 2M;
```

يوضح هذا الأمر بسيط مثل ما تحصل عليه من نتائج. فيقوم بتعريف:

- اسم ملف جداول قاعدة البيانات، الذي يكون في هذه الحالة video_store.
- الملف الذي سوف يزود بالتخزين الحقيقي، وسوف يتم تكوين هذا الملف آليا. ولاحظ أنه عليك وضع أسماء الملفات بين علامات اقتباس منفردة، مثل 'c:\database\test.dbf'.
- حجم الملف، الذي يكون في هذه الحالة 2MB. "يمكنك استخدام م. تعليمات البرمجة إلى ميغابايت وك. لترمز إلى كيلوبايت". تذكر إتاحة مساحة لأي فهارس وكذلك توسيع خلف الامتدادات الاستهلالية للمقطع.

عند قيامك بتكوين ملف جداول، فأنت ليس مقصور على ملف واحد: فيمكنك تضمين ملفات بيانات قدر ما تريد بأمر واحد فقط، مثل:

```
CREATE TABLESPACE video_store
  DATAFILE 'c:\database\test.dbf' SIZE 1M REUSE,
 'c:\database\test2.dbf' SIZE 1M REUSE ;
```

وفي المثال، قمنا بتعريف ملفي بيانات لجدول الملفات. وقمنا أيضاً باستخدام خيار REUSE الذي سوف يسبب أن يقوم أوراكل بإعادة استخدام ملف بيانات لنفس الاسم في حالة وجود واحد بالفعل.

ويعتبر تحديد ملفين مفيد على وجه الخصوص عندما تصبح إمكانية التخزين مقيدة، لأنه يمكنك وضع الملفات على أقراص مختلفة، ومع ذلك لابد أن تتحاشى إجراء هذه العملية لأنها سوف تسبب قلق صيانة على الدوام. وفي الأوقات الماضية، كانت إضافة ملف ثاني إلى ملف الجداول الطريقة الوحيدة لتجنب المشكلة عند ملء ملف البيانات، ومازالت هذه الطريقة باقية البديل المتاح. ولتجنب هذه المشكلة، يمكنك الآن استخدام خيار AUTOEXTEND:

```
CREATE TABLESPACE video_store
                DATAFILE 'c:\database\test.db' SIZE 1M AUTOEXTEND
ON;
```

تحذير على الرغم أن خيار AUTOEXTEND مغري وجذاب إلى أنه طريقة خطرة لتعريف ملفات الجداول لأنه يفقدك التحكم في حجم ملف البيانات. فقد يتسع الملف لاستهلاك كل القرص. وسوف نعرض كيفية الحد من زيادة حجم الملف قريبا في هذا القسم.

والقرار الذي عليك اتخاذه في هذه الحالة هو قدر المساحة التي أنت بحاجة إليها. ولسوء الحظ هذا القرار لا يعد سهلا ولا توجد له إجابة سهلة. ولكنك تحتاج إلى مراجعة عدد الجداول والفهارس التي سوف تقوم بتحديدها لملف الجداول ومتطلبات التخزين المطلوبة. وتكون الحاجة إلى التخطيط بتوسع لأن أي منطقة تخزين تزداد بشكل ديناميكي عند طلب مساحة أكثر تكون أقل كفاءة من التخطيط الجيد من البداية. وفي حالة التعرض لمشكلة منطقة التخزين وتصل في النهاية إلى منطقة تخزين مجزئة، وسوف تكون بحاجة إلى استيراد البيانات وإعادة تكوين ملف جداول بخواص تخزين مناسبة أكثر ثم تقوم بتصدير البيانات ثانية. وسوف يتطلب هذا تعطيل قاعدة البيانات لفترة من الوقت على حسب كمية البيانات الموجودة. وعندما تتزايد قواعد البيانات، من الممكن أن تستغرق ساعات كثيرة لإجراء الاستيراد، لذلك عليك التفكير أولا.

وبمجرد أن تقوم بتحديد حجم جداول البيانات، عليك منح المستخدمين الامتياز لاستخدامها. ويمكنك إجراء ذلك بواسطة أمر ALTER USER لإعطاء حصص نسبية مخصصة أو غير محددة "حق استخدام كمية محددة من تخزين ملف الجداول" إلى المستخدم على الوجه التالي:

```
ALTER USER nick QUOTA 500K ON video_store ;
```

أو

```
ALTER USER nick QUOTA UNLIMITED ON video_store ;
```

أوامر مفيدة

لإدراج كل ملفات جداول التي يمكن الوصول إليها داخل مخطط قاعدة البيانات الخاص بك، يمكنك استخدام الأمر التالي:

```
SELECT tablespace_name FROM user_tablespaces ;
```

وللبحث عن كمية المساحة غير المستخدمة المتروكة في ملف الجداول الخاص بك، استخدم الأمر التالي:

```
SELECT SUM (bytes) FROM dba_free_space
WHERE tablespace_name='TEST';
```

لن يتمكن كل المستخدمون من الوصول إلى عرض القاموس هذا، ولكن يوجد عرض آخر يسمى user_free_space لمراجعة الوحدات في مخطط قاعدة البيانات الخاص بك.

معاملات اختيارية

عند تجربة قاعدة البيانات، سوف يعمل أمر CREATE TABLESPACE جيدا. ومع ذلك، عند إعداد قاعدة بيانات إنتاج، لابد أن تكون حذر من المعاملات الاختيارية التي يمكنك تحديدها. وعلى وجهة التحديد لابد من مراجعة الخيار AUTOEXTEND التالي، الذي يتيح نمو أحجام الملفات داخل حدود وإتاحة نظام التشغيل.

AUTOEXTEND: من الممكن أن يتمدد اسم ملف البيانات الذي يتبعه هذا المعامل في الحجم آليا "حتى يصل إلى حدود نظام التشغيل أو قيمة يمكنك تحديدها بواسطة عبارة MAXSIZE". تتيح لك هذه العبارة تشغيل أو تعطيل هذه الخاصية ويحدد حجم الامتداد عند ازدياد ملف البيانات. ولاحظ أن هذا لا يقوم بمد الحصص.

DEFAULT STORAGE: يقوم هذا المعامل بتحديد معاملات التخزين الافتراضية لأي وحدة يتم تكوينها في ملف الجداول هذا، مثل الجداول والفهارس. وعموماً، سوف تكون بحاجة لتحديد معاملات التخزين لكل وحدة على حدة لأنهم سوف يتضمنوا متطلبات تخزين مختلفة وجامحة، ولكن من خلال هذه العبارة تصبح عملية الإدارة أسهل إذا كان لديك وحدات متعددة بنفس المتطلبات. لاحظ أنه لو قمت بحذف هذه العبارة، سوف تنطبق افتراضات النظام لأوراكل.

ONLINE "افتراضي": سوف يكون ملف الجداول متاح فوراً للمستخدمين.

OFFLINE: سوف يكون ملف الجداول غير متاح مبدئياً للمستخدمين.

PERMANENT "افتراضي": سوف يتم استخدام ملف جداول للوحدات الدائمة، مثل الجداول.

TEMPORARY: سوف يتم استخدام ملف الجداول لتخزين الوحدات المؤقتة فقط، تماماً مثل منطقة العمل الخاصة بالفرز.

REUSE: من الممكن أن تظهر الكلمة الرئيسية هذه بعد عبارة DATAFILE:. إذا قمت بتحديدها، سوف تتم إعادة استخدام الملف الموجود "وإلا يتم تكوين ملف جديد".

تلميح عند إعداد تطبيق حقيقي، سوف تكون بحاجة إلى ملفين جداول متاحين على الأقل لك: الأول خاص بالبيانات والآخر خاص بالفهارس. ضع في اعتبارك أيضا تجزئة مناطق التطبيق المختلفة مثل الموارد البشرية والمحاسبية والهندسية إلى جداول ملفات مختلفة. وإذا قمت بتأجيل ذلك إلى أن تنمو قاعدة البيانات، سوف يكون من الصعب جدا التجزئة فيما بعد. ولنقل الجداول إلى ملف جداول مختلف، عليك إما تصدير أو استيراد الجداول أو استخدام أمر "الاستعلام الفرعي CREATE TABLE AS SELECT". وأي الطريقتين تأخذ كمية وقت هائلة بديلة لنقل وإعادة تكوين الفهارس. وهذا يعني أن هذه الجداول لن تكون متاحة للمستخدمين. فيمكن إجراء ذلك ولكن من الأفضل جدا اتباع البناء الجيد الصحيح من البداية.

والآتي مثال حول استخدام هذه المعاملات لزيادة حجم الملفات الخاصة بك آليا:

```
CREATE TABLESPACE video_store
 DATAFILE 'c:\database\test.db' SIZE 1M REUSE
```

```
AUTOEXTEND ON MAXSIZE 10M
OFFLINE;
```

سوف تلاحظ أن هذا الأمر يبدأ أيضا جدول الملفات الجديد في حالة منفصلة وهذا يعني أنه لن يكون متاح للمستخدمين. ولجعله مباشر، استخدم أمر ALTER TABLESPACE الموضح بالقسم التالي.

أوامر متعلقة

إذا كنت تريد إزالة جدول بيانات لأنك لم تعد بحاجة إليه، على سبيل المثال، أو كنت تريد تحديد ملف أكبر بدل من إضافة ملف منفصل، يمكنك آنذاك حذف ملف جداول بواسطة أمر .

DROP TABLESPACE *tablespace_name* ;

ولإجراء ذلك، عليك التأكد من إزالة أي مقاطع، مثل الجداول والفهارس، ومع هذا إذا كنت متأكد أنك لن تقوم بإزالة أي شئ ضروري، يمكنك استخدام.

DROP TABLESPACE *tablespace_name* INCLUDING CONTENTS;

ملاحظـة عند "إسقاط" ملف جداول، فأنت لا تقوم بحذف أي ملفات ولكن فقط الهيكل الاعتباري المطبق على تلك الملفات. إذا كنت تريد إزالة الملفات بالمثل، سوف تكون بحاجة إلى استخدام أوامر نظام التشغيل.

إذا كنت تريد تغيير خصائص جدول الملفات لإجراء أشياء مثل إضافة ملف بيانات جديد بطريقة مباشرة أو غير مباشرة، فعليك استخدام الأمر التالي:

ALTER TABLESPACE *new_parameters* ;

تحـذيـر إذا قمت بإسقاط ملف جداول، سوف يظهر قيد في أرشيف نافذة التنبيه.

تلميح توجد طريقة واحدة يمكنك الاستفادة منها عند تكوين جداول الملفات وهـــي إعداد جدول ملف واحد للجداول الكبيرة وآخر للجداول الصغيرة. إذا نظرت إلى أي قاعدة بيانات موجودة، من المحتمل أن تجد أنـــك تمتلـك جـداول صغيرة كثيرة، غالبا خاصة حالات وجداول الرموز حيث لن توجـد علـى الإطلاق صفوف عديدة من البيانات. وقد توجد جداول بيانات كبيرة أقـــل. وإذا قمت بتكوين جداول بيانات منفصلـة لكـل نـوع واسـتخدام عبـارة DEFAULT STORAGE مناسبة، يمكنك آنذاك تجاهل معـــاملات تخزيـن الجداول. وبهذه الطريقة،سوف توجد في جداول أي ملفات جداول خـواص تخزين منتظمة. وسوف نتعرض لعبارة STORAGE في القسم التالي.

تكوين جدول

بمجرد أن يتم إعداد ملف الجداول، يمكنك تكوين الوحدات المختلفة لمخطط قاعدة البيانات فيـــه، مثل الجداول والفهارس. وسوف تكون بحاجة إلى الحصول على الامتيازات الكافية لكي تكـــون قادر على تكوين وحدات، تتضمن الحصة. وسوف نناقش الامتيازات في قسم "إعداد الامتيـازات" في هذا الفصل.

وبافتراض أنه توجد لديك الامتيازات، فقد يمكنك آنذاك البدء بتكوين جدول أو أكثر. في هــذا الفصل، سوف نقوم بمناقشة وحدات أوراكل ٧ المتوافقة، وعلى وجه الخصوص الجداول العلاقية بما أنها تتعارض مع جداول الوحدة المتاحة الآن في أوراكل ٨. ولحسن الحظ فقد تـــم تصميـم الجداول العلاقية ككيانات ذات سمات. وإذا كنت غير معتاد على هذه المصطلحات، عليك مراجعة الفصل الخامس، تصميم قاعدة البيانات.

تلميح الجدول العلاقي هو عبارة عن جدول ذات صفوف من أعمدة، ويوجد بكـل عمود نوع بيانات معين. وجدول الكيان بالمثل إلا أن الأعمدة يمكنها احتواء بيانات مركبة أو معقدة. يوجد بعض الأدوات الشيقة في الفيجوال بيسـك ٦ لمعالجة الجداول. وسوف نرى هذه الأدوات في نهاية هذا الفصل.

وفي هذه الحالة تحتاج إلى مراجعة أنواع البيانات التي قمت بتصميمها واتخاذ القرار حـــول إمكانية تنفيذهم في جدول أوراكل.

ملاحظة لقد ازدادت المنوعات لكل أنواع البيانات في أوراكل مع أوراكل ٨، ولكن لا يوجد تعدد للاختيار مقارنة مع المتاح عن طريق موردي قاعدة س بيانات آخرين. ومن الناحية الواقعية إمكانية اختيار محدود يعني إمكانية اتخاذ قرار محدودة الشيء الذي يعد ميزة دائما.

إذا اخترت شراء خيار Objects "الذي يعتبر حزمة اختيارية من أوراكل"، يقوم أوراكل ٨ بإكمال الجداول العلاقية العادية التي تشاهدها في كل RDBMS "نظام إدارة قاعدة البيانات العلاقية" بجداول الوحدة. وبدلا من الاقتصار على أعمدة تحتوي على نوع بيانات منفرد، من الممكن أن تحتوي جداول الوحدة على أعمدة أنواع بيانات معرفة بواسطة المستخدم تشبه الوحدة.

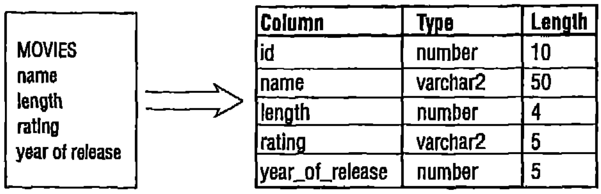
أنواع بيانات أوراكل

يوجد إجراء واضح وهو التفكير بلغة أنواع بيانات الفيجوال بيسك وذلك عند إعداد جدول. وعلى الرغم أن التفكير بهذه الطريقة لن يضر من ناحية المفهوم "سوف تقوم بعد كل هذا بإرجاع هذه الجداول ثانية إلى متغيرات فيجوال بيسك كيفما اتفق"، إلا أنه لا تفضل أن يكون هذا عامل مسيطر. تحتوي الفيجوال بيسك على مجموعة غنية للغاية من أنواع البيانات، بينما تكون أنواع البيانات لأوراكل أكثر تقييدا. ومع ذلك عندما يقوم برنامج الفيجوال بيسك بقراءة البيانات من أوراكل، يمكن تحليل المعلومات في طرق مختلفة. على سبيل المثال، يمكن استخدام عمود الجدول المكون من رقم واحد لتخزين واسترداد متغير بولين.

يتكون الجدول من سلسلة من الصفوف، كل منها يتكون من أعمدة. أوراكل ٧ يحتوي على ٢٥٤ عمود بينما يحتوي أوراكل ٨ على ١,٠٠٠ عمود الذي يعتبر كافي لمعظم المتطلبات الخاصة بك. وعليك اتخاذ القرار أولا حول كيف تريد تحويل الأنواع المصممة حسب السمة إلى أنواع قواعد البيانات المتاحة. وباستخدام تطبيق الأفلام، عليك النظر إلى الكيانات واتخاذ القرار حول أنواع البيانات الأفضل لمناسبة السمات والمساحة المتوقعة لاحتواء قيم البيانات. يعطي الشكل ٦-٢ مثال حول كيفية إجراء هذا على جدول Movies.

الشكل ٦-٢

تحويل الكيانـــات إلى الجداول



تمثل الجداول كيانات التصميم بينما تمثل الأعمدة السمات. يتم الرمز إلى كل مثـــال بكيـان مستقل بصف في الجدول. وأنواع البيانات المتاحة لأوراكل موضحة في الجدول ٦-١.

الجدول ٦-١: أنواع البيانات المتاحة عند تصميم جدول في أوراكل ٧.

نوع البيانات	البيـان
CHAR "الحجم"	سلاسل ثابتة الطول لحجم الطول. ويمكن إعداد هذه الجداول بطول ٢٥٥ بايت، والطول الافتراضي ١.
DATE	التاريخ العام وحقل الوقت
LONG	حقل حرف متغير الطول يصل إلى ٢ ميغابايت
LONG RAW	بيانات ثنائية متغيرة الطول "مثل الرسومات" يصل إلى ٢ ميغابايت في الطول.
RAW "الحجم"	بيانات ثنائية تصل إلى ٢,٠٠٠ بايت في الطول.
NUMBER "حجم"	رقم صحيح مع أرقام قياس
NUMBER "الدقة و التدرج"	رقم الفاصلة العائمة مع أرقام دقة أمام الفاصلة العشرية وأرقام قياس بعد الفاصلة العشرية.
VARCHAR2 "الحجم"	سلسلة متغيرة الطول تصل إلى حجم طول الحروف بحد أقصى ٢,٠٠٠ بايت
VARCHAR "الحجم"	نفس VARCHAR2 ولكننا لا نوصي باستخدامه.

أكثر الأنواع استخداما بين أنواع البيانات هذه NUMBER وVARCHAR2 وDATE. وأنـــواع البيانات الأخرى التي يوجد بها بعض القيود هي سLONG وLONG RAW. ومـــن بيـــن هـــذه القيود:

- لا يمكن وضعهم في فهرس
- من الممكن أن يكون لديك عمود
- يمكن استخدامهم في الاستعلامات الفرعية SELECT أو العبارات مثل WHERE وORDER BY.

بالإضافة إلى أنواع البيانات أعلاه، توجد أنواع بيانات جديدة متاحة لمستخدمي أوراكل ٨. ولقد تم تصميم أوراكل ٨ لقابلية التدرج: فكلما كان كل شئ أكبر كلما كان أفضل. ولقد تم تصميم أنواع البيانات الجديدة لتفي بالاحتياجات الحديثة للملفات الثنائية الأكبر وتتضمن احتياجات الصوت والفيديو والاحتياجات الفضائية واحتياجات أخرى. ويتم التعرض لأنواع البيانات هذه في الجدول ٦-٢.

الجدول ٦-٢: أنواع بيانات إضافية في أوراكل ٨.

نوع البيانات	البيان
BLOB	بيانات ثنائية تصل إلى ٤ جيجابايت
CLOB	بيانات حرفية بعدد واحد بايت حتى ٤ جيجابايت
NCLOB	رموز حرفية بوحدات بايت متعددة "مثل الرموز الأحادية" حتى ٤ جيجابايت
BFILE	مرجع إلى ملف خارج قاعدة البيانات حتى ٤ جيجابايت.

يتم تصميم أنواع البيانات هذه لتناسب الاحتياجات الطارئة لتخزين رسومات كبيرة ووصلات فيديو وأنواع بيانات متعدد وسائط أخرى.

أنواع بيانات Long

إذا دعتك الحاجة إلى الاستفادة بكثرة من أعمدة الجداول الكبيرة، التي تزيد على [illegible] بايت أو الجداول التي تحتوي على بيانات تنوع ثنائي، عليك الانتباه إلى أنه لا توجد طريقة سهلة لوضع البيانات داخل هذه الجداول أو استخراجها منها. وكمطور فيجوال بيسك، سوف تكون بحاجة إلى أن تلجأ إلى طرق GetChunk و AppendChunk التي تزودك بها طرق إمكانية وصولك إلى البيانات. تقوم هذه الطرق بإرسال أو استرداد البيانات في "حزم" وتقوم

بتكرار هذه الطريقة حتى يتم معالجة كل البيانات. ولا يمكنك إرسال هذه البيانات عن طريق إجراء مخزن، وفي الحقيقة، سوف تكون بحاجة إلى العمل مع مجموعات سجلات قابلة للتحديث. والتصميم الجيد يظهر عند سحب الأعمدة الطويلة خارج الجدول كلية ثم وضعهم في الجدول الخاص بهم عن طريق مفتاح مناسب. ويمكنك التفكير حول هذا كعملية تطبيع زائدة، مع وجود بعض الميزات منها أنه لن يتم سحب هذه الحقول الكبيرة إلى استعلام إلا إذا كنت تريد ذلك أو كنت تستطيع معالجة الأعمدة الأخرى بالطرق العادية (ولحسن الحظ باستخدام الإجراءات المخزنة بدل مجموعات التحديث القابلة للتحديث). يمكن الاحتفاظ بهذه الجداول في ملف جداول مختلف.

تكوين جدول في النهاية

وأخيرا أنت على استعداد لتكوين جدول. تذكر من الفصل الثاني أن الجدول هو وحدة قاعدة بيانات في مخطط قاعدة البيانات الخاص بك والذي يمكنك تصوره كجدول إلكتروني. يتكون هذا الجدول من صفوف وأعمدة ما احتواء كل صف على بيان بطبعة منفردة من البيانات، مثل الفلم أو الفيديو في المثال الذي بين أيدينا. يحتوي كل عمود على نوع مختلف من البيانات، كل منها بغرض محدد؛ على سبيل المثال، اسم الفلم أو المدة المحددة له، الذي قمنا بتحديد اسم مناسب له. يمكن ربط هذا المفهوم مباشرة بتكوين جدول أوراكل الخاص بنا.

والصيغة المبسطة لهذا هي:

```
CREATE TABLE table_name (
 column1_name data_type <constraint_information>,
 repeat for each column
) TABLESPACE tablespace_name
STORAGE storage_clause;
```

يتم تحديد كل عمود بالتعاقب ويتبع بنوع البيانات الخاص به واختياريا بعبارة CONSTRAINT. ويتم فصل كل تحديدات العمود بواسطة فاصلات وتكون داخل أقواس هلالية. من الممكن أن تتبع تحديدات العمود بمعاملات اختيارية. تقوم عبارة CONSTRAINT بتحديد نوع البيانات التي يمكن وضعها في أعمدة الجدول والتي سوف يتم اكتشافها في القسم التالي.

ونقوم الآن بأخذ التصميم الاعتباري بكل كيان على حدة ثم نقوم بتحويله كل واحد إلى جدول. ويجب تحويل كل من سمات الكيان إلى عمود وليس من المهم الترتيب الذي توضع فيه الأعمدة ولكن يمكنك وضعهم حسب المصطلح بحيث تكون أعمدة المفتاح في البداية وأي أعمدة تدقيق

مثل Time_Stamp أو User_Name تأتي في النهاية. ولا توجد قواعد لهذا ولكن من الممكن أن يجعل الانتظام في التصميم الحقيقي العمل مع الجداول أسهل بكثير.

وبمجرد أن تفهم نوع بيانات أوراكل المقابل لكل واحدة من السمات الخاصة بك، يمكنك كتابة سكريبت لتكوين الجدول. ومثال السكريبت موضح في شكل التعليمات البرمجية٦-١. ولقد قمت بتبسيط هذا السكريبت الواضح لتكوين جدول Movies عن طريق إهمال أي قيود، ولكنه يغطي أكثر المعاملات الاختيارية المستخدمة عموما. وسوف نقوم بتغطية القيود في القسم المسمى "تحديد القيود".

تعليمات برمجية ٦-١

```
CREATE TABLE movies (
 movie_id  NUMBER(10),
 movie_name  VARCHAR2(100),
 length   NUMBER(4),
 year_of_release NUMBER(4),
 rating   VARCHAR2(6))
TABLESPACE video_store;
```

من التعليمات البرمجية ٦-١، يمكنك مشاهدة أنه من السهل تحديد بعض أنواع البيانات من التصميم الاعتباري "على سبيل المثال، تحتاج الأعمدة الحرفية بمتغير الطول إلى تحديد الحد الأقصى للطول فقط" ولكن قد يكون الآخرين أكثر صعوبة. ومع قيم المفتاح الأساسية، سوف تفهم الحقل الكبير بشكل أكبر. فحتى ١٠ أرقام يسمح بسجلات متنوعة ولكن تذكر أنه من الممكن وجود سجلات بين قيم المفتاح "المستخدمة" التي تم حذفها وكذلك "افتقدت" أرقام المسلسل. يعطيك المفتاح الأساسي أيضا الحرية عند سحب أرقام المسلسل كما تريد بدون القلق من الزيادة على قيمة الحد الأقصى المحددة للمفتاح.

ملاحظة شئ واحد يجب عليك ملاحظته عند العمل مع الفيجوال بيسك كمقدمة أمامية لقاعدة البيانات الخاصة بك وهو أن قاعدة البيانات سوف تنمو ضروريا بشكل أكبر. وإذا قمت بتصميمها لعدد أقل، فقد تريد تطوير تعليمات البرمجة التي تعمل مع متغير صحيح لاحتجاز حقول المفتاح. وعندما تصل إلى أكثر من ٣٢,٧٦٧، سوف تكون في حالة فائض. عليك المحاولة والبدء في أنواع متغير الفيجوال بيسك الطويلة.

المعاملات الاختيارية

على الرغم أن الأمثلة الموضحة في الأقسام السابقة تعد كافية، إلا أنه من المهم تحديد متطلبـاتك التي تريد توافرها في قاعدة البيانات بتفاصيل أكثر. وسوف تعمل هذه الخطـوة علـى تحويـل التصميم الخاص بك من تصميم لا قيمة له إلى آخر بالغ الأهمية. وإذا لم تفكر حـول متطلبـات قاعدة البيانات الخاصة بك، فسوف ينعكس هذا سلبيا على التنفيذ الحقيقي للتصميم.

ولتكوين قاعدة بيانات مصممة جيدا، عليك الاستفادة من المعاملات الاختيارية الخاصة بجملـة CREATE TABLE. وأدناه تجد أهم هذه المعاملات. وإذا لم تكن لديك متطلبات أخـرى، فعليـك التعود على إعطاء مزيد من الاهتمام لمتطلبات التخزين وملف الجـداول حيـث سـيتم تكويـن الجدول.

ملف الجدول: يتيح لك هذا المعامل تحديد ملف الجدول الذي تريد تكوين الجدول بـه. وسوف تتيح لك التقاط ملف جداول خاص بأغراض لتصميم مثل نشر لجـدول علـى مشغلي الأقراص المنفصلة لأغراض الأداء أو ملف الجداول المتعلقة بتطبيق. ومع أنـك لست بحاجة إلى تحديد ملف الجداول "سوف يقوم أوراكل باسـتخدام ملـف الجـداول الافتراضي الخاص بك في حالة إهماله"، إلا أنه يفضل إجراء ذلـك. ومحلـل قـاعدة البيانات الجيد يقوم بتعليم المطورين على أساس اعتبـارات تصميـم قـاعدة البيانـات الأساسية.

التخزين: يتيح لك هذا المعامل تحديد متطلبات التخزين الأولية للجدول الخاص بك التـي على أساسها ينمو هذا الجدول. وإذا حذفت معامل التخزيـن، سـوف يقـوم أوراكـل باستخدام افتراضات ملف الجداول والتي قد لا تتناسب مع متطلباتك. وتوجد تغطية أكثر لهذا المعامل أدناه.

الافتراض: يتيح لك هذا المعامل تحديد القيمة الافتراضية لعمـود إذا لـم تقـم جملـة INSERT بتزويد قيمة. ومن الممكن أن تكون قيمة فعلية "مثل ٢٠٠" ومن الممكـن أن تكون تعبير ولسوء الحظ لا يمكن ربط هذه القيمة برقم مسلسـل. ومـع هـذا، يمكـن استخدام هذا الخيار لوضع اسم المستخدم وختم الوقت لأغراض التدقيق. "يراجع قسـم "ميزات التدقيق" فيما بعد بهذا الفصل".

ملاحظـة عند تكوين جداول، يمكنك أيضا تحديد عبارة AS subquery تسـمح لـك هذه العبارة بتكوين جدول لا يأخذ فقط هيكل العمود الخاص به فيه جملـة SELECT أيضا. وفي هذه الحالة لا تقوم بتحديد أي أعمدة لأن هذا سـوف يتم ضمنا. ومثال هذا موضع في الفصل ٨ "تحديث قاعدة البيانات".

معاملات أخرى متاحة لضبط الأداء وتتطلب فحصهم أعمق حول كيفية استخدام المقاطع. ولمزيد من هذه المعاملات، عليك إلغاء حماية دليل Oracle SQL Reference الذي يشحن مع أوراكل.

عرض معلومات الجدول

للكشف عن أعمدة في أي جدول، أكتب.

```
DESC table_name
```

ولأدراج الجداول الموجودة بمخطط قاعدة البيانات الخاص بك استخدم الأمر التالي:

```
SELECT table_name, tablespace_name FROM user_tables;
```

تحديد معاملات التخزين

لتحدي معاملات التخزين على وجه الدقة، عليك تقييم احتياجات التخزين الحالية الخاصة بك، وتقدير كيفية نمو هذه الاحتياجات، وعليه تقوم بالتخطيط لهذا النمو، عندما تقوم بتكوين قاعدة بيانات لأغراض الاختيار، فقد لا يكون من الضروري تحديد حجم الجداول.والأكثر أهمية أن تكون منتبه لهذا التقدير عند إعداد قاعدة بيانات الإنتاج لأنه سوف تظهر افتراضات كثيرة تعمل جيدا في البداية يرجع أيضا إلى متطلبات التخزين.

معظم الطرق لتقدير مساحة التخزين ينتج عنها الحاجة إلى قدر محدد من الخبرة والمحاولة والخطأ. وإذا كانت لديك القدرة على إعداد قاعدة بيانات أولية بكمية هائلة من البيانات الممثلة، فإنه يمكنك تحديد كمية التخزين التي قمت باستخدامها "بعد مدير مؤسسة أوراكل أداة ممتازة لذلك" ويمكنك التخمين بذكاء جدول كيفية نمو هذه المتطلبات على مدار الأشخاص القليلة القادمة وكذا. وبعد مرحلة البداية، يمكنك إضافة كل متطلبات الجدول وإضافة مساحة إضافية لنمو الحجم غير المتوقع وربما الجدول الجديدة ويكون لديك آنذاك تقديرا أفضل لملف الجدول.

لا ينطبق معامل التخزين فقط على الجداول ويكن أيضا على الفهارس والعناقيد وكذلك مقاطع إلغاء التعديلات واللقطات. يقوم أوراكل بتعيين مساحة في الامتدادات التي تعتبر مجموعات من قطع البيانات المتواصلة. وقبل أوراكل ٧-٣ النصيحة القياسية كانت بتخزين الجدول في أقل أو أكبر الامتدادات. ومع أوراكل ٧-٣ فما بعد لم تعد هذه قضية ذو أهمية قلم يعد لديك الحاجة لتحديد رقم الحد الأقصى لامتدادات. على الرغم أنه من الأفضل أن تكون كل الامتدادات في ملف الجداول بنفس الحجم.

ومعامل التخزين هو في الحقيقة عبارة تحتوي على معاملات اختيارية أخـــرى عديـدة. وإذا كنت تتذكر من الفصل الثاني، بنية قاعدة البيانات لأوراكل أن كل وحدة، مثل الجـــدول، تكـــون مضمنة في هيكل اعتباري يسمى مقطع. كل من معاملات التخزين يعتبر الجدول ٣-٦. معاملات التخزين.

الجدول ٦-٣: معاملات التخزين

المعامل	البيـان
INITIAL	تخصيص التخزين الولي للجدول بالبايت (يمكنك استخدامه للرمز إلى كيلو بايت و م للرمز إلى ميغابايت).
NEXT	حجم الامتداد التالي عندما يزداد الجدول عند الحجم الأولي، بالبايت أيضا.
MINEXTENTS	رقم الحد الأدنى لامتداد الذي يكون موجود للجدول افتراضيا سوف يكون ١ ويراك على أنك قمت بتكوين الامتداد الأول بشكل كبير مناسب.
MAXEXTENTS	رقم الحد الأقصى لامتدادات التي يمكن أن ينم إليها الجدول الخاص بك، يمكنك تحديد رقم أو استخدام الكلمة الأساسية UNLIMITED لتتيح للجدول أن ينمو حسب الحاجة.
PCTINCREASE	في كل مرة يتم فيها إضافة امتداد جديد، يمكنك زيادة حجم أحدث امتداد آليا بالقيمة في المائة المزودة في هذا المعامل. على سبيل المثل، إذا قمت بتحديد قيمة ١٠٠ بالمائة أكبر من القيمة السابقة.

والمثال الموضح لاستخدام كل المعاملات مدرجة في الجدول ٣-٦ موضح فـــي التعليمـــات البرمجية ٦-٢. يتيح لك السكريبت في التعليمات البرمجية ٢-٦ تكوين جـــدول Movies الـــذي يتضمن كل معاملات التخزين.

تعليمات برمجية ٦-٢

```
CREATE TABLE movies (
Movie_id  NUMBER(10),
Movie_name  VARCHAR2(100),
Length   NUMBER(4),
Year_of_release NUMBER(4),
Rating   VARCHAR2(6))
TABLESPACE video_store
STORAGE (
INITIAL 50K
NEXT 50K
MINEXTENTS 1
MAXEXTENTS 4
PCTINCREASE 100);
```

ويحدد تكوين الجدول بهذه الطريقة أن مساحة القرص ٥٠ ك. بايت سوف تكون مخصصة للامتداد الأول من مقطع الجدول. ولأن عدد الحد الأدنى من الامتدادات ١، سوف يتم تخصيص الامتداد الأول فقط وعندما يأتي وقت نمو الجدول خلف الامتداد الأول هذا، سوف يتم تخصيص الامتداد ٥٠ ك بايت للامتداد الثاني. وسوف تكون كل الامتدادات ١٠٠% أكبر من السابقة لها، لذلك سوف نشاهد الثالث الذي يكون ١٠٠ ك بايت والرابع الذي يكون ٢٠٠ ك بايت موضح في الشكل ٦-٣. ويحدد الحد الأقصى بالإمتدادات محدود ب ٤ لذلك لن يتم تكوين أي إمتدادات أخرى. وفي الحقيقة، بالطبع، سوف تستخدم الجداول أكبر من الأحجام الموضحة هنا.

50k	50k	100k	200k
Initial Extent	Second Extent	Third Extent	Fourth Extent

الشكل ٦-٣: أربعة امتـدادات مخصصة لملف الجدولVideo. Store

تلميحات على معامل التخزين

لأقل مساحة من مجزئة القرص، عليك وضع PCTINCREASE = 0 ولاستخدام أفضل، حدد الإمتداد الأولي على قيمة كبيرة وخطط لكي لا يكون لديك أكثر من أربعة أو خمسة امتدادات مراقب الاستخدام (وتعتبر حزمة لأداء لمدير مؤسسة أوراكل، التي تأتي مع خادم مؤسسة أوراكل وتكون متاحة على انفصال، ممتازة لهذا الاستخدام لأنه توجد كل المعلومات عند أطراف أصابعك. وعندما يصبح الاستخدام تام اتخذ القرار حول ما إذا كنت تريد الامتداد أو إعادة التنظيم.

إنها فكرة جيدة أيضا أن تجعل كل الجداول في ملف تستخدم نفس معاملات التخزين. وأسهل طريقة لذلك هي استخدام عبارة STORAGE عند تكوين ملف الجداول بدل من الجدول.

وعلى الرغم أنه يفضل أن يكون لديك امتدادات قليلة، إلا أنه يوجد ميزة قليلة في تحديد حد أقصى صغير لأن هذا سوف يقيد توسع التخزين إذا لم تقم بالتقدير المناسب.

أوامر متعلقة

بمجرد أن تقوم بتكوين جدول، ليس من الضروري تنفيذ العمل الخاص بك. وتوجد مرات عديدة عندما تقوم بتغيير التصميم الخاص بك لسبب ما، وعادة يكون تغيير العمل أو أنك لم تعد في حاجة إلى الاحتفاظ بأنواع محددة من البيانات. توجد أيضا أوامر أخرى مرتبطة ارتباطا وثيقا بجملة CREATE TABLE التي تساعدك في صيانة الجداول. وإذا لم تعد في حاجة إلى جدول، يمكنك حذفه بواسطة:

```
DROP TABLE table_name;
```

يسمح لك أمر ALTER TABLE إضافة أعمدة جديدة أو تغيير نوع البيانات لأعمدة موجودة أو إضافة معاملات اختيارية جديدة.

```
ALTER TABLE ADD (new_column, new_type) ;
ALTER TABLE MODIFY (existing_column, new_type) ;
ALTER TABLE
 new_parameters ;
```

يضع المثال التالي الأمر لاستخدامه:

```
ALTER TABLE ADD (user_name VARCHAR2(30);
```

توقع التغيرات

سوف نعرف أن التغير هو الشيء الوحيد الثابت في هذا العالم، وخصوصا في عالم الكمبيوتر. وفي الأيام الماضية، كانت هذه حقيقة معروفة تماما. وبإجراء التغيرات لمدة ثابتة، كان النظام على حسب الملف مريع، لذا يقوم المصممون غالبا بتكوين سجلات أطول من الحاجة لكي يتيحوا المجال للتغيرات الطارئة. وبالنسبة للتغيرات غير المتوقعة، لم تكن المساحة الإضافية كافية أو كانت في المكان الخاطئ.

ولقد تغيرت الأوقات بشكل مثير للانتباه. ومع أوراكل، يعد من السهل للغاية إضافة عمود إلى جدول. ويمكنك إجراء ذلك مباشرة وقد يكون كل شئ في حالة عمل جيدة "مع أنه لا يوجد ضمان لهذا". ومع ذلك، عليك الانتباه جيدا إلى أنه بينما يعد من السهل إضافة عمود ومن الصعب تغيير آخر، إلا أنه خلف المستحيل إزالة عمود. وعليك إعادة تكوين جدول الشيء الذي سوف يستغرق وقت طويل. ولا يوجد شئ أسوأ من مشاهدة جدول بأعمدة الفكرة التي كانت تعد جيدة ولكنها زالت من لعدم مواكبتها للتغيرات.

تبديل الجداول بالتطبيق

أكبر مشكلة تواجهها عند تبديل الجداول هي عندما تستخدم قاعدة بيانات مباشرة. إذا قمت بتعطيل النظام، فإنه يمكنك الآن إجراء كل التغييرات الخاصة بك ثم اختبارهم بالكامل ثم تقوم بتشغيل النظام ثانية. إذا لم تتمكن من إجراء ذلك، فمجرد تشغيل كل التغييرات الخاصة بك على قاعدة بيانات الاختبار بنجاح، عليك تحديث قاعدة بيانات الإنتاج بأقصى سرعة ممكنة. وإذا قمت بإضافة عمود إلى جدول، سوف يتم تعليم الإجراءات المخزنة التي تشير

إلى ذلك الجدول كغير مقبولة بسبب أنهم لم يقوموا بإعادة التجميع مع أنه يمكنك تنفيذهم.

والمشكلة الأكبر تكون مع جملة INSERT التي سوف تفشل نتيجة للعدد الخاطئ للأعمدة. إذا لم تتمكن من تعطيل قاعدة البيانات لإجراء هذه التغييرات، فعليك الانتباه إلى هذه المشكلة وتقوم بتعديل وإعادة تجميع الإجراءات المخزنة على وجه السرعة.

إذا كان الحد الأمامي للفيجوال بيسك بحاجة إلى تحديث، فسوف تكون لديك مشكلة أكبر سوف تحتاج إلى خطة منسقة للتحديث قاعدة البيانات والإجراءات المخزنة والحد الأمامي وقد تكون بحاجة إلى تعطيل قاعدة البيانات.

من الممكن أن يكون تنفيذ الصيانة للأعمدة الزائدة إجراء ضار. يمكنك تحويل قاعدة البيانات إلى وضع الإنتاج، ولكن قواعد الأعمال بسرعة هذه الأيام لدرجة أنه سوف تكون كمية هائلة من الضغط لعمل تغييرات متصلة بقاعدة البيانات. والحقيقة أن جزء الخاص بالمصمم منا هو مشاهدة تصميم جيد محدد له وقت استقرار.

تلميح يعتبر المصمم الذي يعتبر جزء من الفيجوال بيسك ٦ منتج ممتاز لجعل صيانة الجدول أسهل بشكل أكثر. وسوف نقوم بتغطية هذا المنتج في القسم النهائي من هذا الفصل.

سوف يتم إضافة الأعمدة وغالبا زيادة حجمها. ولكن من السهل إجراء ذلك، فليس من اللازم تحديد حقل مكون من ١٠٠ حرف بينما يقوم ١٠ حروف بنفس المهمة، خاصة إذا كان عمود ثابت الطول. وفي الحقيقة، قد لا يفضل استخدام الأعمدة ثابتة الطول ما عدا عندما يكونوا في حجم صغير.

وعند مدى محدد، عليك إعادة تحديد موقفك تجاه حقيقة الأمر أن الجدول به قرارات بتواريخ قديمة تؤثر على التصميم الخاص به مما يدعوا إلى إعادة تكوينه. وفي هذه الحالة، يمكنك استخدام أمر SQL مثل:

```
CREATE TABLE AS subquery
```

ولنسخ البيانات، فأنت ما زلت بحاجة إلى جدول جديد ثم تقوم بإسقاط الجدول القديم وإعادة تسمية جدول جديد للاسم الأساسي (استخدم الأمر RENAME لهذا). قد تكون بحاجة أيضا إلى استخدام الاستيراد والتصدير لهذا الغرض.

تحذير

يمكنك تعديل جدول أثناء استخدامه. على سبيل المثال، إذا قمت بإضافة عمود انتبه إلى أن أي أجراء مخزن يصل إلى هذا الجدول سوف يتم تعليمه كغير مترجم. وبشكل شيق، سوف يزال أوراكل السماح لك بتنفيذ الإجراء، ولكن عليك استغلال الفرصة حول إمكانية تشغيله أو لا. وفي أوراكل ٧-٣، تقوم الإجراءات بإعادة الترجمة آليا إذا كانوا غير مقبولين؛ تحدث إعادة الترجمة في التنفيذ التالي. وعلى وجه العموم، إذا كان لديك إجراء يحذف صف، فلابد أن يعمل ولكن سوف يفشل الإدخال إذا لم تقم بإدخال القيم الكافية لكل عمود.

تحديد القيود

الخطوة التالية في تكوين الجدول تتضمن القيود. والقيود هي عبارة عن سمات تتيح لك تقييد القيم التي يمكن إدخالها إلى عمود جدول أو أكثر. وتعد هذه القيود الطريقة الأساسية لتحديد قواعد الأعمال الابتدائية.

وباستخدام القيود، يمكنك إعداد قاعدة البيانات الخاصة بك. بذلك لن يتمكن المستخدم النهائي من إدخال البيانات التي لا تعمل وفق القواعد التي تريد التخلي عنها. وليس من المهم سواء كانوا يستخدمون SQL*Plus أو فيجوال بيسك. سوف يكون صاحب الجدول والمستخدم بامتيازات من مسئول قاعدة البيانات القادرين فقط على تخطي القواعد (عادة بواسطة إسقاط أو تعطيل القيود).

يوجد نوعان من القيود: قيود جدول وقيود عمود، مع أنه في معظم الأحوال يقوموا بتنفيذ نفس الغرض، ولا يوجد اختلاف من ناحية التشغيل بين الاثنين. وسوف نراجع كيفية استخدام كلا النوعين، ولكن تذكر دائما أنه ليس من المهم أيهما تستخدم: ولكن كن متوافق.بينهما.

قيود العمود

تشير قيود العمود إلى القيود الموجودة في تحديد عمود الجدول. ويتم تحديد هذه القيود مثل عبارات CONTSAINT التي تأتي بعد أسماء الأعمدة.

إذا نظرت إلى التعليمات البرمجية ٦-٢، سوف تلاحظ أننا لم نقم بتضمين أي قيــود. فــأنت لست بحاجة إلى تحديد اسم القيد، ولكن سوف تحصد نتائج ذلك إذا قمت بذلك. ومع هذا إذا قمـت بذلك، سوف يقوم النظام بتكوين اسم مشفر لهم.

وصيغة عبارة CONSTRAINT كالآتي:

```
CONSTRAINT constraint_name constraint_type
```

وأنواع قيود العمود موضحة في الجدول ٦-٤.

الجدول ٦-٤: أنواع قيود العمود.

اسم القيد	البيـان
CHECK	يحدد تعبير بسيط (الذي قد يتضمن مراجع العمود) الذي لابــد أن يستوفيه كل عمود.
REFERENCES	يحدد أن القيمة الموضحة في هذا العمود يجب أن تظهر أيضا فــي PRIMERARY KEY للجدول المرجع (أو الأصل). يعطـــي هـــذا القيد لقاعدة البيانات التكامل المرجعي المدمج. وهذا يعني أيضـا أن الصف المشار إليه في جدول الأصل لا يمكن حذفه أثنـــاء وجــود الصف في هذا الجدول
NOT NULL	يحدد أن العمود لا يمكن أن يحتوي على NULL (والافتراضي أنــه يمكنه ذلك). ويعتبر هذا القيد ضمني إذا تم تحديـد PRIMERARY KEY أيضا.
PRIMERARY KEY	يعرف العمود بكونه PRIMERARY KEY بمعنى أنـــه فريــد ولا يمكن أن يكون NULL. ويتم تكوين الفهرس آليا في هذا العمود.
UNIQUE	يعرف العمود بكونه فريد، بمعنى أنه لا يمكنك إدخال نفس القيمـــة إلى أكثر من صف واحد. ويتم تكوين الفهرس آليا في هذا العمود.

ومثال استخدام كل القيود المدرجة في الجدول ٦-٤ موضح في التعليمات البرمجية ٦-٣

تعليمات برمجية ٦-٣

```
CREATE TABLE videos (
video_id NUMBER(10)
CONSTRAINT video_id_pk PRIMARY KEY,
movie_id NUMBER(10)
CONSTRAINT video_movie_id_fk REFERENCES movies(movie_id)
UNIQUE,
status VARCHAR2(10) DEFAULT 'available' NOT NULL,
rental_date  DATE,
customer_id  NUMBER(10) DEFAULT 0,
purchase_date DATE NOT NULL);
```

في هذه الحالة يتم تحديد عمود Video_id بواسطة قيد KEY PRIMARY، عمود Video_id مع قيد KEY PRIMARY الذي يشير إلى عمود Movie_id من جدول Movies. توجد أيضا بعض الأعمدة بقيد NOT NULL الذي يعني أنه أينما تحاول إدخال سجل جديد في جدول Videos، يقوم تعيين مفتاح الربط أولا بالفحص للتأكد أنه توجد عمليات موفقة في جدول Movies.

ملاحظة تعتبر عبارة CONSTRAINT اختيارية تماما. على سبيل المثال، يمكنك تعريف القيد في هذه الطريقة 'id number(10) PRIMARY KEY. في هذه الحالة، سوف يقوم أوراكل بتعيين اسم قيد من النموذج SYS _Cnnnn حيث أن nnnn مجرد رقم ما. ومع هذا من الأفضل جدا تضمين الكلمة الرئيسية CONSTRAINT واسم القيد.

قيود الجدول

تقوم قيود الجدول بتنفيذ نفس الوظيفة مثل قيود العمود: فيقوموا بفرض قيود على ما يمكن أن تقوم بإدخاله إلى أعمدة قاعدة البيانات. ومع هذا فإن قيود الجدول سوف تظهر في جملة CREATE TABLE في النهاية بدل من بعد كل عمود. وبهذه الطريقة من الممكن تطبيقهم على الجدول ككل وكذلك الأعمدة المنفردة. ولكن قيود الجدول تختلف عن قيد. والميزة الوحيدة هي أنهم يتيحوا تكوين فهرس مركب بمعنى فهرس منفرد يتضمن تحديده أكثر من عمود واحد.

وأنواع قيود الجداول موضحة في الجدول ٦-٥. وسوف تلاحظ أنه في معظم الحالات يكونوا نفس الشيء مثل قيود الأعمدة ما عدا أنهم يتضمنوا اسم العمود "أو الأعمدة" التي يتطابقوا معها.

الجدول ٦-٥: خيارات تحديد جدول

اسم القيد	البيان
CHECK "الحالة"	يحدد الحالة التي يجب على كل صف المرور بها.
FOREIGN KEY REFERENCES	يحدد العمود "أو الأعمدة" التي يجب أن يكون لها قيد FOREIGN KEY يشير إلى عمود "أو أعمدة" في جدول آخر.
FOREIGN KEY "عمود"	يحدد العمود أو الأعمدة التي يجب أن يكون لها مفتاح أساسي.
UNIQUE "عمود"	يحدد أن الأعمدة يجب أن يكون بها مفتاح فريد.

وكمثال لاستخدام قيود الجدول، انظر التعليمات البرمجية ٦-٤. فقد تم الاحتفاظ بقيود NOT NULL كقيود عمود للتبسيط.

تعليمات برمجية ٦-٤

```
CREATE TABLE movies (
movie_id  NUMBER(10),
movie_name  VARCHAR2(50),
length   NUMBER(3),
rating   VARCHAR2(6),
year_of_release NUMBER(4),
 CONSTRAINT movie_id_pk PRIMARY KEY (movie_id),
 CONSTRAINT movie_rating_chk
 CHECK (rating IN ('G', 'PG', 'PG-13', 'NC-17', 'R')));
```

سوف تلاحظ أن قيود الجدول تظهر في نهاية السكريبت "على الرغم من إمكان ظهورهم في أي مكان" ويت فصلهم بفاصلات. ولقد قمنا بتخصيص قيدين جدول: واحد لتحديد عمود

Movie_id بما أنه يحتوي على مفتاح أساسي ويحدد الآخر قيد CHECK الذي يؤكد في هذه الحالة على أن قيمة العمود التقديرية واحدة من خمس إمكانات.

ملاحظة

لإدراج القيود الموجودة لديك بمخطط قاعدة البيانات، استخدم الأمر التالي:
SELECT constraint_name, table_name FROM
.user_constraints;

تنفيذ التكامل المرجعي

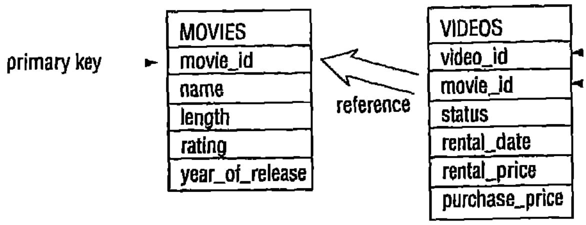
يتم بناء قواعد البيانات العلاقية على العلاقات بين الجداول. توجد أنواع عديدة من العلاقات، ولكن أشهر علاقة هي علاقية أساسي-تفصيل لجدول، على سبيل المثال، Movies. ومثال آخر شهير هو العلاقة بين جداول Invoice-to-Line-Items في قاعدة بيانات الشراء. تظهر علاقات واحد لمتعدد في مخطط علاقة الكيان. ولكل صف أساسي، قد يوجد صفوف تفصيل صفر أو واحد أو متعددة.

عند القيام بتكوين جداول لتمثيل علاقات أساسي-تفصيل، يجب التأكد أن كل سجل تفصيل لا يحتوي على أي بيانات عادية بالنسبة للسجل الرئيسي حيث أن جدول Video لا يحتوي على أي معلومات عن الفلم ذاته، ولكنه يحتوي على معلومات متعلقة فقط بالشريط الحقيقي. وإذا أردت الكشف عن نوع هذا الفلم، يجب ربط جدول التفصيل "الفيديوهات" بالجدول الرئيسي "الأفلام" بواسطة عمود في الجدول التفصيل، فنسمي العمود مفتاح الربط، في هذه الحالة هو Movie_id "راجع الشكل ٦-٤". يشير هذا العمود إلى المفتاح الأساسي في الجدول الرئيسي. وتسمى هذه العملية بالتكامل المرجعي. وفي حالة عدم وجود مفتاح توصيل في جدول الأصل "قد يكون تم حذف مفتاح التوصيل أو أنه غير موجود أساسا".

ولتجنب مثل هذه الحالات، يمكن استخدام القيود لإجراء شيئين:

١- إيقاف إدخال أو تحديث صف التفصيل بدون مفتاح أصل موصل.

٢- إيقاف حذف أي مفتاح أصل في حالة الإشارة إليه بواسطة مفتاح الربط.



الشكل ٦-٤ التكامل المرجعي في حيز التطبيق.

على الرغم أنه من الممكن تنفيذ قواعد التكامل المرجعية هذه في تعليمات البرمجة المزودة، إلا أنه من الأفضل جدا تنفيذهم بواسطة القيود. ويمكن إجراء هذه العملية مرة واحدة. في التعليمات البرمجية ٦-٣، يظهر تحديد مفتاح الربط كالآتي:

```
movie_id NUMBER(10)
 CONSTRAINT video_movie_id_fk REFERENCES movies(movie_id)
```

الذي نقوم فيه بتحديد القيد على العمود movie_id في جدول Videos "المفتاح الأساسي". والقرار الذي عليك اتخاذه هنا هو ما إذا كنت تريد إعطاء أعمدة التوصيل نفس الاسم من عدمه. ومع أن هذا قد يؤدي إلى قليل من الالتباس عند قراءة القيود، إلا أنه يقوم عادة بعمل حالات الربط أسهل عند الكتابة.

وبالتطبيق أينما تقوم الآن بإدخال سجل تفصيل، يقوم الجدول الرئيسى بالقراءة آليا لكشف ما إذا كان مفتاح الأصل موجود من عدمه. وأينما تقوم بالحذف من الجدول الرئيسي، عليك المسح لتوصيل مفاتيح الربط. ولن يقوم المطورين غالبا بوضع فهرس على مفتاح الربط "وهذا الإجراء غير آلي مثل المفاتيح الأساسية". من الممكن أن يؤدي هذا إلى إبطاء في إجراءات الحذف لأنه من المطلوب أن يقوموا بمسح كامل للجدول من جدول التفصيل للتأكد أنه من الممكن حذف الصف الرئيسي. لذلك، كما تشاهد، التكامل الآلي المرجعي لا يأتى بدون ثمن.

تحديد الوضع للفهارس

افتراضيا، أي فهرس مكون بواسطة قيد سوف يتم وضعه في نفس ملف الجداول مثل الجدول وسوف يتلقى المقطع تخصيص تخزين افتراضي. والاعتماد على الافتراضات لا تعد في الغالب حالة دقيقة، ومن الأفضل وضع الفهارس في ملف الجداول الخاص بهم "عادة مع فهارس أخرى" يعتمد التصميم الدقيق على نوع البيانات الموجودة لديك وكيفية استخدامها. ومع هذا يمكنك تحديد ملف الجداول الذي سوف يحوي الفهرس على النحو التالي:

```
CREATE TABLE videos (
 video_id    NUMBER(10)
```

```
CONSTRAINT video_id_pk PRIMARY KEY
 USING INDEX
 TABLESPACE video_store
  STORAGE storage_clause,
movie_id NUMBER(10),
status VARCHAR2(10) DEFAULT 'available' NOT NULL,
rental_date  DATE,
customer_id  NUMBER(10) DEFAULT 0,
purchase_date DATE NO
```

استخدام القيود

سوف تقوم الفهارس بجعل جمل SELECT أسرع بشكل طبيعي ولكن سوف تعمل من الناحية الأخرى على إبطاء الإدخال والتحديث والحذف للبيانات. سوف تقوم قيود أخرى أيضا بإبطاء الإدخال والتحديث لا القيد سوف يكون بحاجة للفحص أولا. وفي حالة التكامل المرجعي، قد يتم إنهاء قراءات البيانات الإضافية لتمام إجراءها، لذا عليك توظيف القيود بعناية. وإذا لم تكن متأكدا، ضع القيود في أهم الجداول الخاصة بك. ثم، عند الاطمئنان على البيانات التي تم إدخالها، يمكنك إزالة بعض القيود من أكثر الجداول المثقلة لديك ثم يمكنك مشاهدة التأثير على الأداء.

وعندما يقوم قيدي PRIMARY KEY و UNIQUE بتكوين فهرس آليا، سوف تجد أن هذا سوف يزيد من وقت حفظ البيانات لأنك تقوم بحفظ البيانات إلى الجدول كما يكون المفتاح إلى الفهرس. ولتلطيف تأثير هذه العملية، عليك التأكد أن الفهارس موضوعة ملف الجداول على قرص منفصل. ويمكنك إجراء ذلك عن طريق خيار USING INDEX من عبارة CONSTRAINT أو، بالتبادل، بأمر CREATE INDEX.

إذا كنت تقوم بتحميل المعلومات بالجملة، قد تريد تعطيل القيود أو حتى إسقاطهم. يمكنك إجراء ذلك بواسطة أمر ALTER TABLE كالآتي:

```
ALTER TABLE DISABLE constraint_name
```

أو

```
ALTER TABLE DROP constraint-name
```

وهذا شيء واحد على أن إعطاء أسماء ذو معنى للقيود الخاصة بك سوف يكون له ثمنه

تكوين الفهارس

الفهرس هو وحدة قاعدة بيانات تتيح لك الوصول مباشرة إلى أي سجل معين في جدول. وبالنسبة لأي جداول بها سجلات متعددة، سوف يقوم الفهرس بتحسين وقت الاسترداد. والطريقة البديلة للبحث عن سجل هو إجراء مسح كامل للجدول الذي يستغرق دقائق أو أكثر على جدول كبير وهذا لن يجعلك شهير جدا بين زملائك في العمل. يحتوي الفهرس على إدخال لكل قيمة في العمود "أو الأعمدة" المفهرس، يفضل أن يكون الفهرس فريد. يمكن تأسيس فهرس على أساس عمود واحد أو قد يكون فهرس مركب قائم على أعمدة متعددة. هذه الأعمدة ليس بالضرورة أن تكون حقول متجاورة في الجدول. قيم NULL غير مخزنة في فهرس.

ويعتبر وضع الفهرس بدقة ضروري بالنسبة للأداء. تذكر أن كل الفهارس سوف تفرض غرامة أداء صغيرة على الإدخالات أو التحديثات لأنه سوف تكون الحاجة إلى أن يقوم أوراكل بقراءة قطع عديدة لإيجاد السجل الذي تريده. إذا كان الجدول صغير جدا أو كنت دائما في حاجة إلى قراءة كل الجدول، ففي هذه الحالة لا يفضل استخدام الفهرس.

وأسهل طريقة لتكوين فهرس هي أن يكون لديك فهرس مكون آليا بواسطة قيد مثل PRIMARY KEY أو UNIQUE. عقليا سوف تقوم بإعداد ملف جداول واحد على الأقل على مشغل قرص منفصل مخصص فقط للفهارس. والنتيجة هي احتواء أقل لإمكانية الوصول إلى القرص مقارنة مع قراءة الفهرس والبيانات من نفس القرص "بافتراض أنه غير موجود بالفعل في الذاكرة".

تلميح إذا كنت تقوم بتصدير بيانات جملة، كما يكون الحال عندما يكون لديك دفعة تحديثات أو أن تقوم باستيراد بيانات إلى جدول، سوف يكون التحميل أسرع إذا قمت بإسقاط الفهرس وإعادة تكوين فهرس بعد الانتهاء. ولقد أضافت هذه الخطوة ميزة التأكد من تنظيف الفهرس الشيء الذي عليك محاولة إجراءه بانتظام وبأي طريقة.

ولتكوين فهرس، استخدم جملة SQL التالية:

```
CREATE [UNIQUE] INDEX index_name ON
 table_name (column1, column2.)
 [TABLESPACE tablespace_name]
 [STORAGE storage_clause];
```

على سبيل المثال:

```
CREATE UNIQUE INDEX video_index ON
 videos (video_id, movie_id)
 TABLESPACE movie_indexes ;
```

ملاحظة للكشف عن نوع الفهارس التي عليك استخدامها، اتبع SQL التالي:

حدد index_name, table_name FROM user_indexes;

يوضح هذا المثال سبب أهمية تسمية الفهارس بأسماء ذو معنى. سوف يساعدك أمر SQL هذا على البحث عن كل الفهارس الخاصة بك، بما فيهم تلك الفهارس المكونة نتيجة شروط القيد.

تكوين واستخدام المسلسلات

في مايكروسوفت أكسس، يوجد حقل Counter، وفي خادم SQL، توجد خواص التطابق الخاصة بالعمود. كلا حقل Counter وخواص التطابق يزيدا من قيمة المفتاح وقتما تقوم بإدخال صف جديد إلى الجدول. ومع هذا لا يقوم أوراكل بتنفيذ العمود الذي تم تكوينه بواسطة المفتاح في أي نموذج. يزود أوراكل بالمسلسلات التي تقوم بتوليد عداد متزايد آليا لا يمكن ربطه بأي جدول أو عمود، وعليك إجراء هذا الجزء بنفسك. هذا يعني أنك لابد أن تبذل جهد وفير لإدخال قيمة المفتاح الفريد الخاص بك إلى الجدول. وعلى الرغم أن المسلسلات تعتبر مرنة "غير مقتصرين على وظيفة واحدة وهي تزويد مفتاح فريد"، إلا أنه توجد إيجابيات قليلة أخرى. ومن هذه الإيجابيات أن توليد المفتاح الفريد لا يسبب أي مشاكل إقفال وكذلك يعتبر أكثر كفاءة من حقل Counter.

تكوين مسلسل

عملية تكوين مسلسل سهلة جدا، وهاك الطريقة:

CREATE SEQUENCE *sequence_name*

عند تكوين مسلسل، يكون الافتراضي توليد الأرقام حتى ٢٠ في الذاكرة الوسيطة بحيث عندما تكون جاهز لاستخدامهم، يكون واحد منهم متاح بشكل طبيعي في الذاكرة ولا تكون بحاجة لتوليده. يمكنك تغيير عدد الإدخالات إلى الذاكرة الوسيطة بواسطة استخدام المعاملات الاختيارية

الموضحة أدناه. يصبح جانب التأثير للذاكرة الوسيطة واضح عندما تكون في حاجة لإعادة بـــدء قاعدة البيانات، فيتم فقد الذاكرة الوسيطة وتوليد مجموعة جديدة من الأرقام.

ولكى يكون رقم المسلسل مفيد لك، لابد أن تكون قادر على إيجاد القيمة الحالية له والحصـول على القيمة التالية المتاحة. ويمكنك إجراء هذا عن طريق وظيفتان مدمجتان:

```
sequence_name.NEXTVAL
```

الذي يقوم بإرجاع رقم المسلسل التالي:

```
sequence_name.CURRVAL
```

الذي يقوم بإرجاع رقم المسلسل الذي تم استرداده بواسطة الدورة الحالية.

هذه الوظائف ليست جمل ولكن لابد من استخدامها كجزء من جملة SQL. وأكـــثر اســتخدام شائع للمسلسل هو توليد فهرس فريد الذي يمكن تكوينه صراحة في جملة SELECT أو ضمنا في جملة INSERT يمكن تنفيذ هذه الوظائف في إجراء مخزن أو كجزء من جملة SELECT التي تم تنفيذها من الفيجوال بيسك. وأمثلة نوع الاستخدام هي:

```
SELECT my_sequence.NEXTVAL INTO next_sequence FROM dual;
INSERT INTO movies VALUES (
 next_sequence.
and
INSERT INTO movies VALUES (
 my_sequence.NEXTVAL, 'Movie Name',
);
```

والجدول المزدوج الذي يتم استخدامه في الأول من هذه الأمثلة متاح لكل المستخدمين وما هو إلا جدول وهمي ويستخدم غالبا في هذا النوع من الجمل التي لا يتم لك الوصول فيـــها إلـــى أي بيانات حقيقية من عمود الجدول "ويسمى كل من NEXTVAL وCURRVAL أعمدة وهمية".

مشاكل الوصول إلى المسلسلات

عندما تتمكن من الوصول إلى رقم المسلسل، يكون هذا الرقـــم فريـــد ومتـــاح فقـــط في الدورة الخاصة بك. ولا يمكنك الحصول علـــى القيمـــة الحاليـــة للمسلســـل فـــي الدورة حتى تقوم باســـترداد القيمـــة التاليـــة. وإذا حـــاولت، ســـوف تحصـــل علـــى رسالة الخطأ التاليـــة:

ERROR: ORA-08002: sequence MY_SEQUENCE.CURRVAL لم يتم تحديده بعد في هذه الدورة.

على الرغم أنه لن يقوم مستخدم آخر بتوليد لنفس الرقم من نفس المسلسل، إلا أنه لا يمكنك مشاهدة الرقم الحالي للمستخدم الآخر. وبناءا على التطبيق الخاص بك، قد يسبب هذا بعض المشاكل. على سبيل المثال، كان لدي تطبيق به شاشة بآخر قيم الصفوف من جداول قليلة. وكان لابد أن تكون البيانات حالية، وعليه كان على خيار واحد تجديد العرض كل ٣٠ ثانية. تضع هذه الخطوة حمل متوازن على الخادم لأنه كان يوجد آلاف عديدة من الصفوف التي كان من اللازم قراءتها دوريا لإنتاج العرض، حتى لو لم يتم تغيير البيانات.

والتقريب الأفضل كان وجود مؤشر ما في قاعدة البيانات إلى أن البيانات قد تغيرت ثم يقوم مزود الفيجوال بيسك بالتسجيل لرقم واحد فقط حتى يتم تغيير البيانات. وأكثر مؤشر واضح أن الصفوف تم إدخالها هو ازدياد رقم المسلسل، ولكن يمكنك البحث عن رقم مسلسل الدورة الخاص بك.

وكان الحل هو حفظ رقم المسلسل في جدول ذو صف واحد في كل مرة يتم فيها توليد رقم مسلسل جديد. وأفضل طريقة لإجراء هذه الخطوة مع الإجراء المخزن. وبقبول هذه الإضافة البسيطة، سوف يكون آخر رقم متاح لكل المستخدمين. ولقد قمت أيضا بتعديل الإجراءات المخزنة التي قامت أيضا بتحديث وحذف الصفوف بما أنهم أحدثوا زيادة أيضا في المسلسل. يوجد لديك أيضا بلايين من الأرقام الفريدة متاحة لذلك استخدام قليل منها لهذا الغرض لن يحدث خطأ.

يقوم الآن مزود الفيجوال بيسك بالتسجيل لكل رقم واحد هذه، وعندما يتغير يعلم أنه قد حان الوقت لتجديد العرض. ولقد قمت بتضمين أرقام مسلسل قليلة في الجدول ذات الصف الواحد هذا المراقبة كل أنواع الأشياء، وقام هذا بتحويل التطبيق من كونه بطيء إلى مستخدم ذو كفاءة لموارد قاعدة البيانات.

معاملات اختيارية

عندما تقوم بتكوين مسلسل، توجد معاملات اختيارية متعددة يمكنك تضمينها بواسطة أمر. على الرغم أنك سوف تجد عادة أن الافتراضات جيدة، لابد أن تكون منتبه لهذه المعاملات في حالات خاصة. يسرد الجدول ٦-٦ المعاملات الاختيارية المتاحة لك عند تكوين مسلسل.

الجدول ٦-٦: المعاملات الاختيارية لتكوين مسلسل.

المؤهل	البيان	القيمة الافتراضية
INCREMENT BY	٢٠	١
MINVALUE x	قيمة الحد الأدنى للمسلسل ١٠٢٦	-١٠^٢٦
NOMINVALUE	افتراضي قيمة الحد الأدنى للمسلسل هي ١ لأعلى و-١٠^٢٦ لأسفل	١ لأعلى و-١٠^٢٦ لأسفل
MAXVALUE x	قيمة الحد الأقصى للمسلسل +١٠^٢٦	+١٠^٢٦
NOMAXVALUE	افتراضي قيمة الحد الأقصى للمسلسل+١٠^٢٦ لأعلى و-١ لأسفل.	+١٠^٢٦ لأعلى و-١ لأسفل
STARTWITH x	القيمة الأولى للمسلسل بعد تكوينها. تعتمد على الزيادة.	تعتمد الزيادة
CYCLE	NOCYCLE	NOCYCLE
CACHE x	يقوم بتوليد أرقام المسلسل x التالي وتخزينهم في ذاكرة وسيطة على استعداد لإمكانية الوصول السريع.	٢٠
NOCACHE	لم يتم وضع أرقام مسلسل في الذاكرة.	
ORDER	يتم توليد المسلسلات وفقا لوقت الطلب.	NOORDER

لن تقوم في الغالب بإعداد مسلسلات، لذا من الأجدر بك أن تقوم بإجراء ذلك جيدا وأن تستخدم أكبر قدر ممكن من المعاملات الاختيارية المساعدة في ذلك ولابد أن يكون لديك وعي بالكمية التي تريد وضعها في الذاكرة الوسيطة (أكبر عدد ممكن إذا كان يتم استخدام هذه المعاملات الاختيارية بشكل ثابت)، وعليك دائما تحديد رقم البداية. ومثال استخدام المعاملات الاختيارية هو:

```
CREATE SEQUENCE my_sequence
  STARTWITH 1000
```

```
INCREMENT BY 1
CACHE 10;
```

يقوم هذا المثال بتكوين مسلسل جديد بادئا من ١٠٠٠٠ ويزداد بمدل واحد كلما تم استخدام NEXTVAL ويقوم أوراكل بتوليد وتخزين ١٠ قيم جاهزة للمستخدم الحالي.

وعليك محاولة توفيق حجم الذاكرة الوسيطة مع عدد مرات استخدام المسلسل. على سبيل المثال، في حالة وجود دفق مدخلات إلى الجدول، عليك إعداد الذاكرة بشكل أعلى وليكن ١٠٠. ومن ناحية أخرى، إذا كان يتم استخدام المسلسل لجدول، مثل جدول الشركة الذي يتم تحديث مرة كل يوم، عليك إذا استخدام خيار NOCACHE.

استخدام المسلسلات في بيئة الاستنساخ

لن تقوم في الغالب باستخدام زيادة أكبر من واحد ولكن توجد حالة يكون فيها تلك ذو قيمة. في حالة استخدام استنساخ متزامن "بمعنى قاعدتين تزامن أو أكثر"، تكون الحاجة إلى التأكد ما إذا كان اثنين من المستخدمين يقوما بإدخال سجل جديد في قاعدة البيانات في نفس الوقت مع عدم وجود تضارب في قيم الفهرس المستخدمة. وطريقة تجنب هذه المشكلة مع قاعدتي بيانات هي عمل مسلسل واحد يبدأ بـ ١ والآخر يبدأ بـ ٢. وإذا كان معامل INCREMENT BY معد على ٢ سوف يقوم جدول منهما بتوليد أرقام مماثلة والآخر أرقام وترية. يمكنك، بالطبع، تطبيق هذا على أي رقم من قواعد البيانات.

صيانة المسلسل

عند استخدام المسلسلات، لابد أن تقبل حقيقة وجود فجوات أحيانا بين قيم المسلسل الذي تقوم باستخدامه. إذا قمت باستخدام هذه المسلسلات للمفاتيح وتم حذف صف، فلن يتم إعادة استخدام رقم المسلسل المستخدم لهذا الصف. وعند إعادة بدء قاعدة البيانات، سوف تختفي كل القيم التي كانت موجودة في الذاكرة الوسيطة للمسلسل ويتم تكوين ذاكرة جديدة. وفي هذه الطريقة "تفقد" أعداد كثيرة. هذه الفجوات لا تسبب أي مشاكل فنية بالفهارس، ومع ذلك يغضب بعض المطورين لهذه الأرقام "المفقودة". قد تظهر المشاكل فقط إذا كنت تستخدم المسلسل لكي تقوم بتوليد الأرقام التي تملي هناك بعدم وجود فجوات (مثل الفواتير). وعليك تطوير نظام توليد الرقم في هذه الحالة.

قد تريد أحيانا إعادة وضع رقم المسلسل مثلما تقوم بإجراء دورة من التجارب على قاعدة البيانات الجديدة. وسوف تبدأ ثانية من ١؛ ولإجراء هذا، عليك حذف المسلسل وتكوين مسلسل آخر.

عليك ملاحظة الأمرين التاليين لصيانة المسلسل:

DROP SEQUENCE sequence_name; الذي يتيح لك حذف المسلسل المسمى وALTER SEQUENCE sequence_name new_parameters الذي يتيح لك استبدال مسلسل موجود بمعاملات جديدة مثل استبدال قيمة الحد الأقصى. لاحظ أنه لا يمكنك استبدال القيمة الحالية ما لم تقم بحذف وإعادة تكوين المسلسل.

للكشف عن المسلسلات التي قمت بتكوينها، استخدم SQL التالية:

NOTESELECT sequence_name FROM user_sequences.

لا يمكنك الكشف عن كيفية تطابق المسلسلات على الجداول بواسطة النظر إلى قاعدة البيانات، لذا حاول الاحتفاظ بهم لأغراض خاصة.

المسلسلات في التطبيق

بمجرد أن يكون لديك رقم فريد، سوف يترتب عليك اتخاذ القرار بدقة حول كيفية استخدامه. وتوجد مجالات مختلفة وعديدة من التفكير حول كيفية الأرقام الفريدة.

مسلسل واحد يناسب الكل

يمكنك تكوين مسلسل واحد واستخدامه لكل شئ وهذه هي الطريقة الأبسط لأنه يوجد لديك مسلسل واحد عند الصيانة وعند تزويد قيم جديدة كما تناسب هذه الطريقة جيدا التصميم حسب الوحدة التي يعتبر فيها كل جدول نوع مختلف من الوحدات وتقوم كل وحدة بمشاركة السمات، مثل رقم واسم المعرف ويحتمل خواص أخرى. يمكن الاحتفاظ بكل هذه السمات في جدول بيان الوحدة "التي تظهر عند التطبيق مثل جدول مثال من الرموز".

وإذا كنت قد قمت بالعمل في نظام مستخدما مسلسل واحد لكل شئ من قبل، قد يصعب عليك أحيانا إدراك ما يحدث وقد تكون دائما بحاجة إلى ربط جدول إضافي لتمام إجراء كل شئ.

مسلسلات متعددة لكل جدول

من الممكن بالتأكيد استخدام أكثر من مسلسل واحد لكل جدول حيث يمكنك تقييد كل واحد بمدى، ثم يمكن لهما التعايش جانبا إلى جنب. والداعي إلى ذلك أنه قد يوجد بقسم مدى "وليكن ١,٠٠٠ إلى ١,٩٩٩" وقسم آخر بمدى مختلف "وليكن ٢,٠٠٠ إلى ٢,٩٩٩". ومع هذا قد يكون من الصعب أحيانا تبديد الخرافة القائلة بأن هذا الإجراء يمكن المستخدمين من فرز المعلومات بصورة أسهل جدا.

وفي الحقيقة، من الممكن أن يكون تحديد مدى في الاستعلام أكثر صعوبة من اختيار رقم قسم محدد في الاستعلامات حتى إذا كان الرقم موجود في الجدول. إذا لم يكن الرقم موجود هناك، فعليك إذا التأكد من وضع أكثر من واحد لكل مجموعة لتدعيم التصميم. وطريقة وضع أكثر من مجموعة معلومات "في هذه الحالة يكون رقم المفتاح ورقم القسم" في عمود يؤدي إلى تصميم سيئ ويجب تحاشيه بكل الطرق. وسوف يتضح هذا عند إجراء الصيانة.

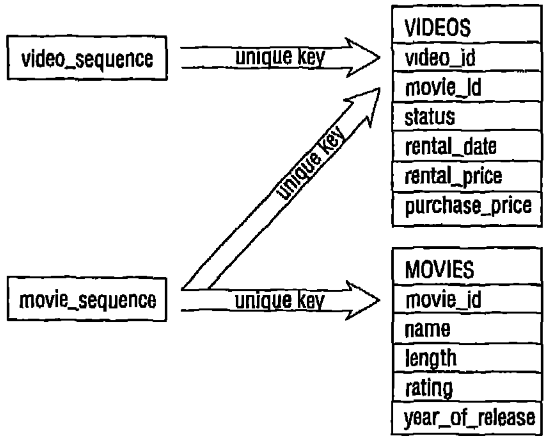
مسلسل خاص بكل جدول

تعبر الطريقة المعتادة لمعالجة المسلسلات هي استخدام مسلسل منفصل للمفتاح أساسي لكل جدول "راجع الشكل ٦-٥". ويظهر هذا الإجراء ملائم مع معظم تصميمات قاعد لبيانات. فمن وجهة المستخدم قد يكون من الأسهل إذا لم استخراج البيانات بطريقة مفهومة ولكن عليك اتخاذ الحذر عند إدخال سجلات جدد.

وعليك التأكد دائما من استخدام المسلسل الصحيح مع الجدول الملائم. وإذا لم يتم ذلك سوف تواجه فيما بعد مشاكل عند محاولة استخدام المفاتيح غير الفريدة. وإذا كانت مقيد باستخدام الإجراءات المخزنة لأي جمل خاصة بلغة معالجة البيانات "INSERT, UPDATE وDELETE"، فعليك وضع هذه المشكلة في أدنى حد لها. شئ آخر عليك إجراءه هو التأكد أنك تستخدم اصطلاح تسمية قياسي للمسلسل مثل اسم الجدول بالإضافة إلى الكلمة "مسلسل" "على سبيل المثال "movie_sequence"".

وعلى الرغم من مشاكل الصيانة التي قد توجهها مع العديد من المسلسلات، إلا أنك تجد المسلسلات أكثر قبولا وأفضل في الطبيعة. فأغلبية الميزات تجعل المسلسلات أكثر جدارة.

الشكل ٦-٥
الطريقة المفضلة لاستخدام المسلسلات.



مسارات التدقيق

والآن قد تعلمنا كيفية تكوين جدول، وبنا لمشاهدة كيف يمكن تعزيز التنفيذ لإعطاء الإمكانية لمسار التدقيق. يوجد بأوراكل إمكانية تدقيق مضمنة يمكن الوصول إليها بواسطة أمر AUDIT SQL، ولكن هذه الإمكانية مصممة خصيصا لتتبع ما تم تنفيذه بواسطة الجمل أو ما تمت محاولته على حدة مخطط قاعدة البيانات. على سبيل المثال، سوف يساعدك التدقيق في تتبع كل العمليات التي تم تنفيذها في الجدول مثل من الذي قام بتبديل الجدول أو إدخال صفوف جديدة أو حذف أو تحديث الجدول. وبما أن هذا يعتبر مفيد لمعرفة الفاعل، إلا أن هذه لا تعتبر تجربة تدقيق بمعنى كف تتبع كيف تم تحديث تلك البيانات أو أي الصفوف تم تحديثها أو حذفها ومع أي البيانات. يقوم التدقيق في أوراكل بتسجيل الجمل التي تقو بالتغيير بدل من القيم الفعلية التي تغيرت. ولتتبع تغييرات القيم، عليك تكوين نظام التدقيق الخاص بك.

ويعد من الهام جدا لك معرفة من الذي يقوم بأي إجراء في البيانات الخاصة بك. ويكن السيناريو بأنك تأتي إلى العمل وتكتشف حذف العديد من صفوف البيانات في واحد من الجداول الخاصة بك، وتعرف عدم وجود داعي لذلك. وعلى الرغم أنه يتم إدخال وحذف الصفوف على الدوام، فمن المسئول عن هذه الحركات ومتى؟ إذا تم إدخال بيانات غريبة، من الفاعل؟ وما هي القيم السابقة، بمعنى أين محاولة التدقيق؟

يوجد المشهد الأول من تدقيق الحركات عادة في كل جداول قواعد البيانات الهامة. ففي بعض البيئات، يعتبر تتبع مسار التدقيق إلزامي. لذا فما هي أفضل طريقة لإجراء ذلك؟ أسهل طريقة هي إضافة حقلين في نهاية كل جدول لاحتواء اسم المستخدم الذي يقوم بتحديث السجل وكذلك طبع الوقت لتسجيل وقت إجراء التغيير. سوف تكون الأعمدة الجديدة هذه User_Name وTime_Stamp. والسكريبت الخاص بتكوين الجدول الجديد كما في التعليمات البرمجية ٦-٥.

تعليمات برمجية ٦-٥

```
CREATE TABLE movies (
 movie_id   NUMBER(10)
 CONSTRAINT movies_movie_id_pk PRIMARY KEY,
 movie_name        VARCHAR2(100) NOT NULL,
 year_of_release   NUMBER(4) NOT NULL,
 rating            VARCHAR2(6) NOT NULL,
 acquisition_date  DATE NOT NULL,
 user_nameVARCHAR(30)
 DEFAULT USER NOT NULL,
 time_stamp        DATE
 DEFAULT SYSDATE NOT NULL)
TABLESPACE movies;
```

يجب أن تكون دقيق عند إدخال القيم إلى هذه الحقول أينما تم إدخالهم أو تحديثهم. يمكنك محاولة إجراء هذه الخطوة في جمل INSERT ولكن سوف تكون بحاجة إلى الاعتماد على مطورين الذين يقومون بكتابة رموز المزود على نفس المواصفات. يجب آنذاك وضع قيم افتراضية خاصة بالأعمدة من USER وSYSDATE كما في التعليمات البرمجية ٦-٥. في هذه الحالة، إذا أغفل مطور إعطاء قيم لهذه الأعمدة، فإن هذه الأعمدة تمتلئ بالقيم آليا.

وأفضل طريقة لإجراء هذا على وجهه الصحيح هي إدخال صفوف جدول من داخل الإجراء المخزن وهذه هي الطريقة الأولى الموصى بها وسوف نغطي هذه الطريقة في الفصل الثامن. أما الآن فنقول أنه يجب استخدام الإجراءات المخزنة لكل جمل لغة معالجة البيانات:التي تستخدم قيم USER ووظائف SYSDATE.

تلميح

عند إضافة أعمدة User_Name وTime_Stamp ، يمكنك الآن معرفة آخر مستخدم قام بتغيير السجل. ولقد شاهدت نظم تجارية كان فيها هذا مستبعد بقدر مسار التدقيق. ولإجراء عمل أفضل، يوجد خياران: إما أن تكتب زناد (إجراء مخزن يتم تنفيذه عند حدوث تغيير محدد في الجدول) ثم احفظ البيانات القديمة في جدول خاص أو أنه يمكن تكوين تعليمات البرمجة الخاصة بنا للكتابة إلى أرشيف الحركة.

وحدات أخرى لمخطط قاعدة البيانات

يوجد ثلاث وحدات أخرى لمخطط قاعدة البيانات التي يجب أن التعرف عليها قبل الانتقال إلى موضوعات أخرى، مع أنه يمكنك الاستمرار بدونهم. وهذه الوحدات هي: العناقيد والعروض والمترادفات. بينما تعد العناقيد طريقة مختلفة لتخزين جدول أو أكثر، إلا أن العروض والمترادفات لا يعدان وحدتان حقيقيتان من حيث الاحتواء على مقطع بيانات ذات بيانات حقيقية متعلقة بهم. ومع هذا فهما وحدتان "ظاهريتان" بمعنى أنهما يشيرا إلى وحدات أخرى. والسبب الرئيسي لهذا هو إخفاء مصدر البيانات عن المستخدمين لأغراض السرية وكذلك لإتاحة الوصول إلى البيانات بشكل أبسط لأغراض الأداء.

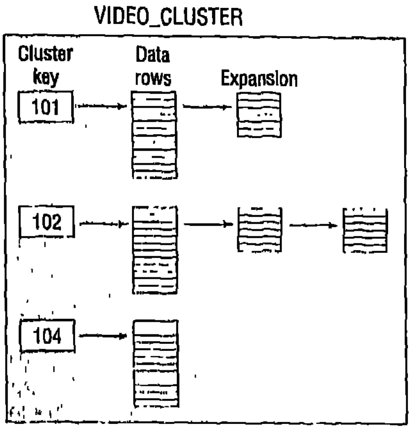
ونوع الحالة التي قد تسبب لك مشكلة هو عندما تريد إعطاء إتاحة وصول لمستخدمين إلى بعض أجزاء الجدول وليس كل الأجزاء؛ على سبيل المثال، قد يكون لديك جدول خاص يتضمن أيضا مرتبات الموظفين وتريد أن يتوصل المستخدمين إلى بعض الجدول "الاسم، المعرف، تاريخ البدء"، ولكنك تريد في نفس الوقت إخفاء المرتبات، فهذا هو الغرض من استخدام وحدة العروض كما في هذه الحالة. من الممكن أن تتمكن تعليمات البرمجة الأمامية والنهائية من إخفاء البيانات السرية، ولكن سوف يتيح هذا المجال إلى المستخدمين القيام بكتابة تعليمات البرمجة للوصول الخاص بهم أو حتى القيام باستخدام استعلام قاعدة البيانات. والطريقة الوحيدة التي يمكن أن يقوم فيها تعليمات البرمجة لفيجوال بيسك بتكرار هذا بدون العرض هي استخدام بعض من الأساليب المتقدمة التي سوف نقوم بمراجعتها في الفصول الأخيرة.

العناقيد

تعتبر العناقيد طريقة بديلة لتخزين بيانات في جدول. وبدل من وجود جدول واحد لكل مقطع بيانات، تتيح لك العناقيد أن يكون لديك جدول أو أكثر في نفس المقطع والأهم من ذلك، يمكنك التحكم في طريقة تخزين الجداول حقيقة في المقطع. والغرض الرئيسي لإجراء هذا هو أنه يمكنك تحديد النظام الذي يتم فيه تخزين الصفوف حقيقة، ثم أنه إذا قمت بتحديد البيانات من المقطع في هذا النظام، فلابد أن يكون أسرع.

يوجد نوعان من العناقيد: عناقيد علاقة مبعثرة وعناقيد مفهرسة. والاختلاف أن العناقيد المفهرسة تستخدم فهرس "راجع الشكل ٦-٦" بينما عناقيد علاقة مبعثرة تستخدم لوغاريتمات مبعثرة للبحث عن الصفوف المخزنة في المقطع.

كلا نوعين العناقيد يتطلبا مفتاح على العنقود وهذا يجعل الجدول الحقيقي يرتبط معك بشكل فعال في حالة تخزين أكثر من جدول. ومع أن هذه تعد ميزة غامرة، إلا أنك سوف تشعر بالجزاء عند إدخال أو تحديث بيانات. والعناقيد لا تبدو أنها تزود بتحسين أداء أفضل كثيرا كما تتوقع، ويمكنك أن تأخذ العناقيد هذه كطريق للاكتشاف عندما يكون لديك وقت فراغ للتسلية.



الشكل ٦-٦
تخزين البيانات حقيقة بواسطة مفتاح العنقود

العروض

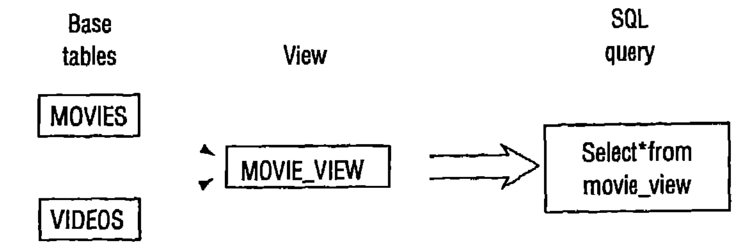
حتى الآن قمنا بتكوين جداول كمخزن للبيانات. ولإمكانية الوصول إليهم، تحتاج إلى وصول SELECT إلى كل جدول، وعليك دائما تنفيذ جدول ربط متطور لإمكانية الوصول إلى البيانات. والبديل لهذا هو تكوين عرض أو جدول "ظاهري" مشتق من أعمدة محددة في جدول حقيقي أو أكثر.

ولتكوين عرض، يجب أن يكون لديك نفس الامتيازات في كل الجداول التي تؤلف العرض كما تريد في العرض الخاص بك. ومع هذا سوف يحتاج المستخدمين النهائيين إلى امتيازات في نفس العرض ولكن ليس على نفس جداول القاعدة للعرض.

والجملة الأمثل لتكوين عرض هي:

```
CREATE VIEW video_view
 AS SELECT movie_name, rating, status
 FROM movies, videos
 WHERE videos.movie_id = movies.movie_id;
```

وميزة العرض هي أن المستخدم ليس لديه أي فكرة عن جداول القاعدة أو، كما في الحالة أعلاه، إذا كان توجد أعمدة مفقودة. إذا كان الاستعلام الفرعي AS SELECT يستخدم أكثر من جدول واحد، فعلى المستخدم عدم الضيق من شروط الربط، فهذه الشروط تكون صريحة في العرض (راجع الشكل ٦-٧).



الشكل ٦-٧ طريقة عمل العرض.

المترادفات

تعتبر المترادفات طريقة بديلة للوصول إلى وحدة. ويمكنك التفكير فيها كبديل للوحدة، سواء كان جدول أو عرض أو إجراء مخزن أو وحدة أخرى. ويعتبر تكوين المترادف سهل:

```
CREATE SYNONYM films FOR movies;
```

والمترادف الذي تقوم بتكوينه "الأفلام" يعتبر خاص ومتاح فقط لهؤلاء المستخدمين الذين لديهم امتيازات فقط. ويفترض أوراكل أن الوحدات التي قمت بتكوين المترادف لها "في هذه الحالة الأفلام" موجودة في مخطط قاعدة البيانات الخاص بك. والأكثر إفادة هي المترادفات العامة المتاحة لكل المستخدمين:

```
CREATE PUBLIC SYNONYM films FOR movies;
```

العروض والمترادفات الخاصة [illegible]

لقد ظل [illegible] ODBC [illegible]
التشغيل ODBC [illegible]
سوف تحصل على أفضل دعم من [illegible]
سوف تتعرض لها في الفصل ١٨ [illegible] OLE

إعداد الامتيازات

لقد غطى هذا الفصل جمل SQL عديدة التي سوف تحتاج امتيازات لها قبل أن تستطيع استخدامهم والتي قد تحتاج إلى نقلها إلى مستخدمين آخرين. يوجد نوعان أساسيان من الامتيازات التي يمكن منحها: امتيازات نظام وتتيح لك تنفيذ جمل SQL محددة وامتيازات وحدة التي يتم منحها إلى وحدة قاعدة بيانات معينة، مثل الجدول.

ملاحظة تعد السرية عنصر حيوي لتشغيل قاعدة البيانات؛ ولعرض مفصل أكثر، عليك مشاهدة الفصل الرابع، إدارة قاعدة البيانات.

امتيازات النظام

سوف يتم منح هذه الامتيازات بواسطة مسئول قاعدة البيانات إلى المطور. وصيغة هذه الامتيازات:

```
GRANT system privilege TO user | role;
```

وعينة امتيازات النظام التي تكون متاحة لك مدرجة في الجدول ٦-٧. ويوجد أكثر من ٨٠ امتياز نظام في الكل.

والآتي مثال لاستخدام الامتيازات المدرجة في الجدول ٦-٧:

```
GRANT CREATE SESSION TO rjames;
GRANT CREATE TABLE TO rjames;
```

الجدول ٦-٧: امتيازات النظام المطلوبة لقاعدة البيانات المثال.

الامتياز	البيان
CREATE SESSION	يتم طلب هذا الامتياز لكي تكون قادر على الدخول إلى قاعدة البيانات.
CREATE TABLESPACE	القدرة على تكوين ملف جداول.
CREATE TABLE	لتمكين المستخدم من تكوين جدول في مخطط قاعدة البيانات الخاصة به. يجب أن يكون لديك حصة كافية في ملف الجداول.
CREATE ANY TABLE	لتمكين المستخدم من تكوين جدول في أي مخطط قاعدة بيانات.
INDEX	القدرة على تكوين فهرس على جدول في مخطط قاعدة البيانات الخاص بك.
CREATE ANY INDEX	القدرة على تكوين فهرس على أي جدول في مخطط قاعدة بيانات أي مستخدم آخر.

الجدول ٦-٧: امتيازات النظام المطلوبة لقاعدة البيانات المثال.

الامتياز	البيان
CREATE SEQUENCE	القدرة على تكوين مسلسل في مخطط قاعدة البيانات الخاص بك.
CREATE ANY SEQUENCE	القدرة على تكوين مسلسل في مخطط قاعدة بيانات لمستخدم آخر.
CREATE VIEW	القدرة على تكوين عرض في مخطط قاعدة البيانات الخاص بك.
CREATE ANY VIEW	القدرة على تكوين عرض في مخطط قاعدة بيانات لمستخدم آخر

امتيازات الوحدة

سوف تكون امتيازات الوحدة الموضحة في الجدول ٦-٨ بشكل عادي هي التي تقوم بنقلها إلى المستخدمين كمطور لكي يمكنهم العمل على مخطط قاعدة البيانات. تذكر أنك قمت بتطوير قاعدة البيانات الخاص بك في مخطط قاعدة بيانات التطبيق التي لا يمكن المستخدمون الآخرون من مشاهدته ما لم تمنحهم الحق في إجراء ذلك.

الجدول ٦-٨: امتيازات وحدة النموذج.

الامتياز	البيان
SELECT	القدرة على إجراء استعلام SELECT أو القدرة على تطبيق وزيادة مسلسل.
INSERT	القدرة على إدخال بيانات إلى الجدول أو العرض.
UPDATE	القدرة على تحديث البيانات في الجدول أو العرض.
DELETE	القدرة على حذف البيانات في الجدول أو العرض.

امتيازات الوحدات الأخرى هي: ALTER وINDEX وREFERENCES وEXECUTE وALL.

والآتي هو مثال حول كيفية إعطاء امتيازات وحدة قاعدة بيانات Movie التي يتم فيها إعطاء المستخدم rjames الامتياز لتنفيذ أمر SELECT وINSERT مقابل جدول Movie.

```
GRANT SELECT ON movies TO rjames;

GRANT SELECT ON movies TO rjames;
```

تلميح يعتبر منح الامتيازات واحدة من أهم الطرق الرئيسية لتأكيد على أن قاعدة البيانات ذو سرية بقدر ما يمكن أثناء قيام المستخدمين بمراجعة ومعالجة البيانات. وكما سوف نشاهد في الفصول المخصصة للإجراءات المخزنة، فإن أفضل طريقة للتأكيد على تكامل وسرية البيانات هي إجراء أي تحديث قاعدة بيانات من خلال الإجراءات المخزنة. وعند استخدام الإجراءات المخزنة، سوف تحتاج إلى إعطاء امتيازات EXECUTE إلى الإجراءات المخزنة بدل من امتيازات INSERT وDELETE وUPDATE في الجدول.

الامتيازات في حيز التنفيذ

يعتبر استخدام آليات السرية المضمنة في أوراكل ضروري لتكوين قاعدة بيانات ذات سرية، ولكن ليس من الضروري أن ينتج عنها نوع من التطبيقات ترغب في مشاهدته. على سبيل المثال، إذا قام المستخدم النهائي بتشغيل واحد من تطبيقات النهاية الطرفية وبعض من الأشياء التي يتم المحاولة فيها يؤدي إلى رسائل خطأ لأوراكل، فهذا سوف يظهر أقل في الجودة. لذلك بينما تكون في حاجة إلى سرية، كيف تقوم بإضافتها بحيث تندمج كليا داخل التطبيق الخاص بك؟

والتوجيه الذي قمت باتباعه هو تكوين جدول آخر لاحتجاز مستخدمي التطبيق. يقوم هذا الجدول باحتجاز المستخدمين الصالحين للتطبيق الخاص بك بخلاف أي شخص له حساب بأوراكل، وتضمين بعض التفاصيل، مثل أسمائهم، ولكن الهدف الرئيسي منه هو أن يحتوي على مؤشرات التي لها يمكن استخدام واجهات التطبيق

وأبسط حل هو تكرار الامتيازات في الجداول، على سبيل المثال، إذا لم يكن للمستخدم الحق في تغيير جدول محدد، سوف يقوم برنامج الفيجوال بيسك بإزالة خيار UPDATE من القائمة أو أن يعطي رسالة لبقة أنه غير متاح لهم أخذ هذا الإجراء.

ويأخذ هذه الإجراء خطوة للأمام، يمكنك تكوين نوع السرية الذي لا يمكن تكوينه بطريقة أخرى. على سبيل المثال، إذا كان لديك جدول يحتوي على بيانات لا يمكن تغييرها بعد نهاية يوم العمل بدون تخويل خاص، فعليك تضمين هذه الإمكانية كعلامة في جدول المستخدم الخاص بك.

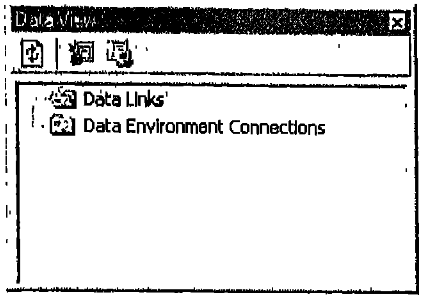
ونوع الجدول الذي قمت باستخدامه هو:

```
CREATE TABLE app_users (
user_name VARCHAR2(30)
ONSTRAINT app_users_pk PRIMARY KEY,
first_name VARCHAR2(20) NOT NULL,
last_name VARCHAR2(20) NOT NULL,
admin_privilege NUMBER(1) DEFAULT 0,
schedule_privilege NUMBER(1) DEFAULT 0);
```

استخدام مصمم قاعدة البيانات لمايكروسوفت

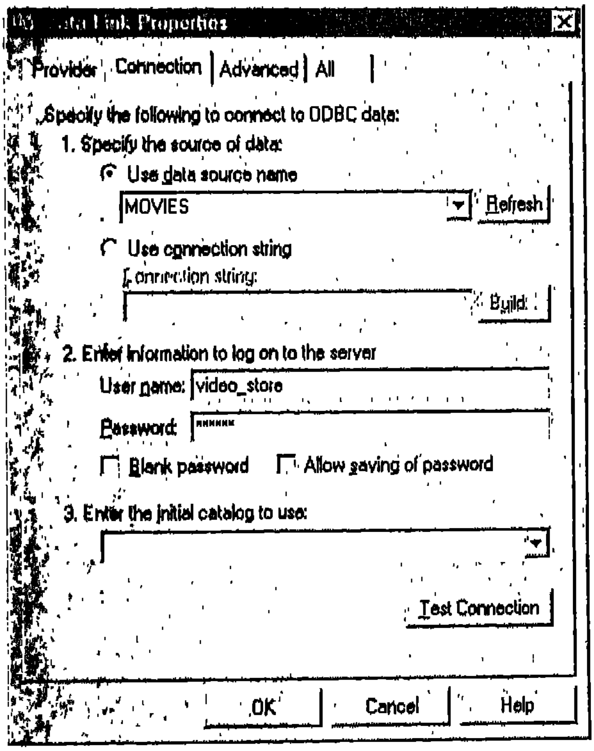
من الضروري فهم الأساليب الضرورية لتكوين الجداول والوحدات الأخرى لمخطط قاعدة البيانات لقاعدة البيانات، ومع هذا فإن الأساليب التي تمت تغطيتها حتى الآن قد تطلبت الكتابة في النص داخل سطر الأوامر الموجه بواسطة SQL*Plus أو تكوين سكريبت بواسطة محرر. ومع إصدار فيجوال بيسك ٦، Enterprise Edition، يوجد لدينا الآن Database Designer الذي يتم تشغيله داخل بيئة الفيجوال بيسك. من الممكن أن يأخذ Database Designer كثير من الكتابة نتيجة تكوين وصيانة الجداول، على الرغم أنك ما زلت بحاجة إلى معرفة ما الذي تقوم بإجرائه وما هي الثغرات الناجمة.

والخطوة الأولى التي عليك القيام بها هي تكوين Data Link "ربط البيانات". ويعتبر Data Link وحدة مستقل عن مشروع الفيجوال بيسك ولكنه يظهر في أي مشروع تقوم بفتحه. عليك ثانية فتح Data View بواسطة تحديد إطار Data View من قائمة View. وسوف تشاهد إطار مثل الموضح في الشكل ٦-٨ بدون Data Link وكذلك بدون Data Environment Connections "وصلات بيئة البيانات".



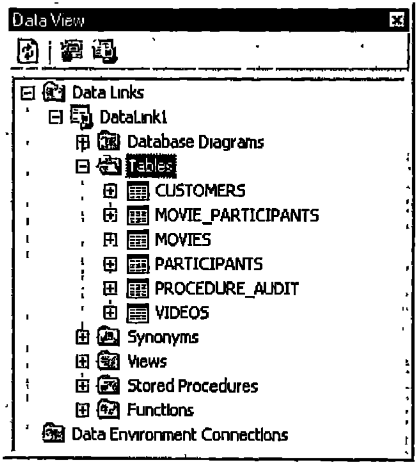
الشكل ٦-٨ إطار بدء Data View.

ولتكوين Data Link جديد، انقر بزر الماوس الأيمن فوق مجلد Data Link ثم اختر Add Data Links من القائمة المنسدلة وسوف تشاهد مربع حوار الخصائص Data Link. حدد مزود OLEDB لمايكروسوفت الخاص بـ ODBC Drivers في الصفحة الأولى. وأدخل ODBC Data Source Nameواسم وكلمة المرور للمستخدم الخاص بك في الصفحة الثانية (راجع الشكل ٦-٩. وقد تريد اختبار الوصلة الخاصة بك بواسطة النقر فوق الزر المخصص. وبافتراض أن الوصلة صالحة، فإنه يمكنك النقر فوق الزر OK ومشاهدة Data Link الجديد في إطار Data View. ويسمى الربط Data Link 1 ولكن يمكنك إعادة تسميته.



شكل ٦-٩ الصفحة الثانية من مربع الحوار لخصائص Data Link.

وإذا قمت الآن بتوسيع شجرة Data Link فسوف تكون مقيد بكلمة مرور، وسوف تشاهد إطار Data View مع كل الوحدات المتاحة عند أطراف أصابعك. في الشكل ٦-١٠ يمكنك مشاهدة أن بند Tables تم توسيعه ليوضح كل الجداول التي لك حق الوصول إليها.



شكل ٦-١٠ يجعل Data Link البيانات مرئية من بيئة الفيجوال بيسك.

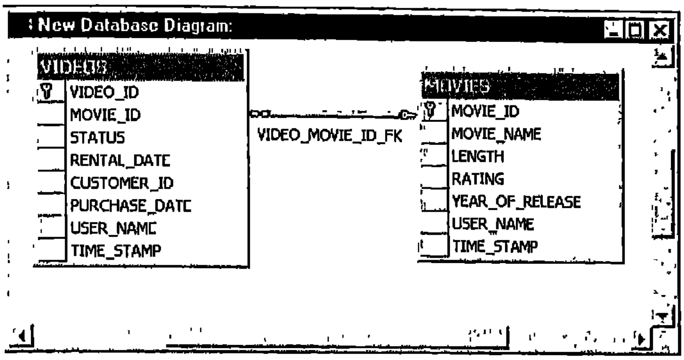
ودعنا نضع أيدينا على نتائج ما نحن عليه الآن. يتيح لك Database Designer الوصول إلى أنواع متعددة من وحدات مخطط قاعدة البيانات لقاعدة البيانات. وفي هذا الفصل، سوف نتعرض لتخطيطات قاعدة البيانات "Database Diagrams" وخيارات Tables. سوف يقوم الفصل التاسع، مقدمة إلى PL/SQL بتغطية الإجراءات والوظائف المخزنة بتفاصيل أكثر.

تخطيطات قاعدة البيانات

يعتبر خيار Database Diagram طريقة لعرض جداول قاعدة البيانات الخاصة بك رسومياً. إنه يزود لك بعرض قد تكون شاهدته مع مايكروسوفت أكسس مع أنه يمكنك تطبيق نفس هذه الإمكانية على قواعد بيانات أوراكل. أولا، عليك النقر بزر الماوس الأيمن فوق Database Diagram من القائمة المنسدلة.

كيف يمكنك وضع جداول في تخطيط قاعدة البيانات الجديد الخاص بك؟ وهذه العملية هي ببساطة مسألة سحب وإسقاط؟ حدد الجدول الذي تريده لإضافة التخطيط من إطار Data View ثم قم بإسقاطه على التخطيط. وفي الشكل ٦-١١، يمكنك مشاهدة أنه تم إسقاط جدولين داخل التخطيط: الأفلام والفيديوهات. يقوم النظام آليا بتحديد ما إذا كان يوجد أي علاقات بين الجدول مستخدما قيود مفتاح الربط ثم قم برسم الربط. انقر بزر الماوس الأيمن على التخطيط، ثم حدد

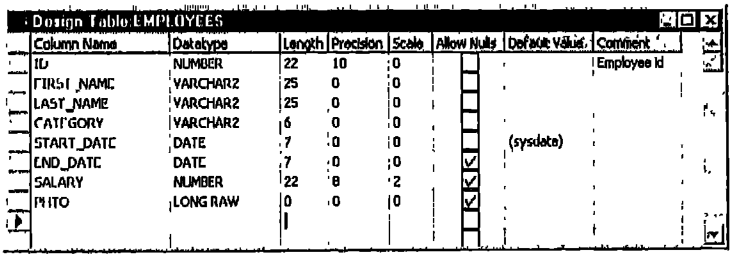
Show Relationship Labels لعرض قيد العلاقة، الذي يكون في هذه الحالة VIDEO_MOVIE_ID_FK. يمكنك الآن إضافة المتبقي من الجداول ثم حظ التخطيط وطبعه.



شكل ٦-١١ تخطيط قاعدة البيانات في طور التجميع.

تكوين الجداول

تعتبر خاصية Database Diagram طريقة جيدة لتوثيق العلاقات بين الجدول بمجرد تكوينهم. ولجعل مرحلة التكوين أسهل، يمكنك استخدام الثاني من أدوات التصميم. انقر بزر الماوس الأيمن فوق مجلد Table في إطار Data View ثم حدد New Table من القائمة المنسدلة. حدد اسم جديد للجدول. والآن املأ مواصفات الأعمدة للجدول الجديد الخاص بك. وفي الجدول ٦-١٢، جدول جديد "Employees" يوجد ثماني أعمدة في حالة مجهزة ويمكن القيام بالتنفيذ.



Design Table:EMPLOYEES

Column Name	Datatype	Length	Precision	Scale	Allow Nulls	Default Value	Comment
ID	NUMBER	22	10	0			Employee Id
FIRST_NAME	VARCHAR2	25	0	0			
LAST_NAME	VARCHAR2	25	0	0			
CATEGORY	VARCHAR2	6	0	0			
START_DATE	DATE	7	0	0		(sysdate)	
END_DATE	DATE	7	0	0	✓		
SALARY	NUMBER	22	8	2	✓		
PHOTO	LONG RAW	0	0	0	✓		

شكل ٦-١٢ تكوين جدول قاعدة البيانات في طور Design.

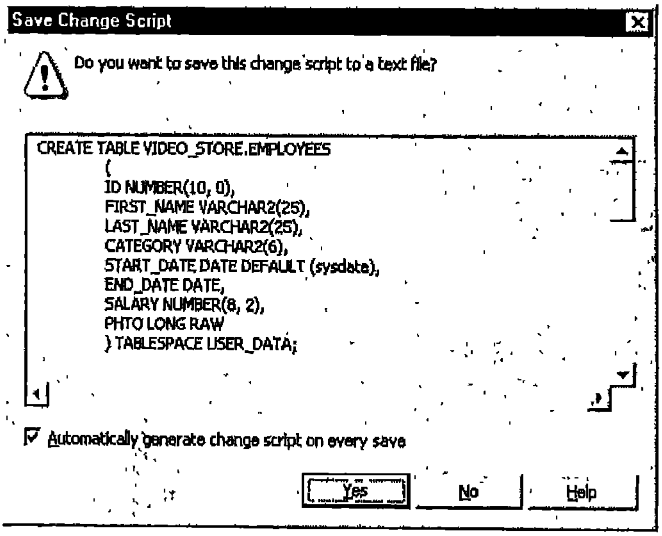
ملاحظة عندما تقوم بتكوين جدول، لاحظ أن الأعمدة الرقمية معطاة طول ٢٢. وهذا لا يتناسب مع طول العمود بأوراكل. ويعتمد هذا على الإنتاج حسب طريقة استرداد ODBC وOLE-DB للمعلومات من أوراكل.

وقبل حفظ الجدول الجديد، عليك إعداد خصائص قليلة. انقر بزر الماوس الأيمن على الجدول في طور التصميم هذا، ثم حدد Properties من القائمة المنسدلة. يمكنك وضع خصائص مفيدة وعديدة مثل:

- تعريف قيد الاختبار "مثل القيد الموضوع على عمود المعدل في جدول Movie".
- تعريف ملف جداول الذي يجب على الجدول الدخول فيه.
- تكوين فهارس على الجدول.

عندما تكون مستعد لحفظ هذا الجدول في قاعدة البيانات، سوف تفاجأ، فسوف يتم إعطاء خيار حفظ السكريبت الذي يقوم بإنشاء الجدول لملف السكريبت. يوضح الشكل ٦-١٣ مربع الحوار حيث يتم من خلاله طلب هذه المسألة. لاحظ أنه يوضح لك سكريبت SQL وتعد هذه أداة تعلم ممتازة.

شكل ٦-١٣ حفظ سكريبت تكوين الجدول.



تحذير إذا نقرت نقرتان مزدوجتان على واحد من الجداول في إطار Data View، سوف تنتقل إلى طور العرض بدل من طور التصميم؛ سوف يتم عرض الجدول في الإطار مع كل الأعمدة لكل الصفوف. ولابد من تجنب هذا على وجه التحديد على أي جدول كبير.

حذف عمود

من أفضل خصائص Database Designer أنه يمكنك حذف عمود من جدول موجود. وقد تلاحظ أننا لم نوضح كيفية إجراء ذلك في هذا الفصل وذلك لعدم إتاحة الفرصة لذلك. وإذا دعتك الحاجة لإجراء ذلك، سوف تجد أنها عملية متعددة الخطوات. ومن الأسهل جدا أن تترك العمود المنتهي أينما هو يكون.

وعندما تكون في طور Database Designer Design، يمكنك تمييز العمود وضغط زر الحذف. يقوم Designer بفعل كل العمل الشاق خلف المشاهد وبتكوين سكريبت لك بقوم بتكوين جدول مؤقت وتستخدم تعليمات البرمجة PL/SQL لنسخ البيانات إلى ذلك الجدول، ثم يقوم بعد ذلك بتكوين جدول جديد مع كل القيود في موضع. والسكريبت الخاص بحذف عمود Salary موضح في التعليمات البرمجية ٦-٦. يوضح السكريبت كمية العمل التي قمت بحفظها الآن لنفسك، وقد لا يعد الطريقة الأفضل لإجراء ذلك، ولكن تعطي هذه العملية السكريبت للتعديل كما تريد:

تعليمات برمجية ٦-٦

```
CREATE TABLE VIDEO_STORE."Tmp_EMPLOYEES"
(
ID NUMBER(10, 0),
FIRST_NAME VARCHAR2(25),
LAST_NAME VARCHAR2(25),
CATEGORY VARCHAR2(6),
START_DATE DATE DEFAULT (sysdate),
END_DATE DATE,
PHTO LONG RAW
) TABLESPACE USER_DATA;
LOCK TABLE VIDEO_STORE.EMPLOYEES IN EXCLUSIVE MODE NOWAIT;
DECLARE
```

```
CURSOR datacursor IS SELECT ID, FIRST_NAME, LAST_NAME,
CATEGORY, START_DATE, END_DATE, PHTO FROM
VIDEO_STORE.EMPLOYEES;
datarecord datacursor%ROWTYPE;
BEGIN
OPEN datacursor;
LOOP
    FETCH datacursor INTO datarecord;
    EXIT WHEN (datacursor%NOTFOUND);
    INSERT INTO VIDEO_STORE."Tmp_EMPLOYEES"(ID, FIRST_NAME,
LAST_NAME, CATEGORY, START_DATE, END_DATE, PHTO) VALUES
(datarecord.ID, datarecord.FIRST_NAME, datarecord.LAST_NAME,
datarecord.CATEGORY, datarecord.START_DATE,
datarecord.END_DATE, datarecord.PHTO);
END LOOP;
END;
DROP TABLE VIDEO_STORE.EMPLOYEES CASCADE CONSTRAINTS;
RENAME "Tmp_EMPLOYEES" TO EMPLOYEES;
ALTER TABLE VIDEO_STORE.EMPLOYEES
MODIFY ID NUMBER(10, 0)
CONSTRAINT NL_EMPLOYEES_4 NOT NULL;
ALTER TABLE VIDEO_STORE.EMPLOYEES
MODIFY FIRST_NAME VARCHAR2(25)
CONSTRAINT NL_EMPLOYEES_3 NOT NULL;
ALTER TABLE VIDEO_STORE.EMPLOYEES
MODIFY LAST_NAME VARCHAR2(25)
CONSTRAINT NL_EMPLOYEES_2 NOT NULL;
ALTER TABLE VIDEO_STORE.EMPLOYEES
MODIFY CATEGORY VARCHAR2(6)
CONSTRAINT NL_EMPLOYEES_1 NOT NULL;
ALTER TABLE VIDEO_STORE.EMPLOYEES
MODIFY START_DATE DATE DEFAULT (sysdate)
CONSTRAINT NL_EMPLOYEES NOT NULL;
CREATE INDEX VIDEO_STORE.IX_EMPLOYEES_ID ON
VIDEO_STORE.EMPLOYEES
```

```
(
ID
) TABLESPACE VIDEO_STORE;
COMMENT ON COLUMN VIDEO_STORE.EMPLOYEES.ID IS
'Employee id';
```

تلميح سوف تلاحظ أنه عند استخدام Database Designer، لا يمكنك إدخال عبارة STORAGE. ولا يعد هذا سيئ كما يبدو لأنه يمكنك إعداد عبارة DEFAULT STORAGE لملفات الجداول الخاصة بك.

خلاصة

لقد قمنا في هذا الفصل بتغطية كل عناصر تنفيذ قاعدة البيانات من التصميم الاعتباري. ويتضمن هذا إعداد ملفات الجداول لاحتجاز وحدات قاعدة البيانات الخاصة بك ثم تكوين جداول وفهارس وقيود أخرى. ثم قمنا بتغطية تكوين واستخدام المسلسلات لإعطاء مفاتيح فريدة للجداول الخاصة بك. وبعد ذلك تناولنا الحقول الإضافية للجداول لإتاحة نظام التدقيق الأولي وكذلك قمنا بتغطية الامتيازات التي أنت في حاجة إليها ليس فقط لتكوين وحدات قاعدة البيانات ولكن أيضا للإتاحة لمستخدمين آخرين استخدامهم. وأخيرا، تعرضنا لأهم الإمكانات الموجودة في Database Designer الذي يعتبر جزء من الفيجوال بيسك ٦.

والآن يمكنك تكوين قاعدة بيانات كاملة، وسوف ننتقل إلى مشاهدة كيفية استرداد البيانات بواسطة استعلامات SQL ثم كيفية عمل تغييرات في البيانات في قاعدة البيانات.

الجزء الثانى

الاستعلامات والإجراءات المُخزنة

- الاستعلامات
- تحديث قاعدة البيانات
- مقدمة عن PL/SQL
- تكوين الإجراءات المُخزنة

٧

الاستعلامات

هناك شيئين يسترعوا انتباه مطور فجوال بيسك عند التعامل مع قاعدة بيانات أوراكل: وضع البيانات في قاعدة بيانات أو استرداد البيانات منها. ويعتبر الإلمام الشامل بالطريقتين شيئا حيويا لتوفير تطبيق مستخدم يمكنه الوصول إلى البيانات المطلوبة بأقصى سرعة ممكنة بدون التحميل الغير ضروري على خادم قاعدة البيانات. ويقوم هذا التطبيق بجلب الحد الأدنى من البيانات التي يمكن من خلالها القيام بالعمل.

سنركز في الفصل ٨ على تغيير البيانات في قاعدة البيانات، ولكن في هذا الفصل سنقوم بتغطية استرداد البيانات من قاعدة بيانات أوراكل. وسنقوم بتغطية:

- مقدمة لـ SQL
- أنواع أوامر SQL
- أمر SELECT الأساسي
- وظائف المجموعة
- عبارة GROUP BY
- تحديد البيانات من أكثر من جدول واحد
- الاستعلامات فرعية
- رفع كفاءة الاستعلامات إلى أقصى درجة
- تحديد مخطط قاعدة البيانات في الاستعلامات

عرض إجمالي لـ SQL

إذا كنت قد تعاملت من قبل مع قاعدة بيانات علاقية. مثل Microsoft SQL Server، فأنت على علم إذن بـ Structured Query Language (لغة الاستعلام الهيكلية)، أو SQL. وSQL هي ناتج بحث قام به الدكتور E. F. Codd في مركز IBM في عام ١٩٧١، وانتهجت شركات قواعد البيانات هذا المفهوم تدريجيا، ولكن أوراكل كانت أول من وفر لغة الاستعلام الهيكلية في الأسواق. وتم تطوير SQL لتصبح قياسية على يد American National Standards Institute (ANSI) (وهو معهد أمريكي لعمل القياسات)، وهكذا تبناه معظم موردو أنظمة قواعد البيانات العلائقية. والقياس الحالي لـ SQL هو ANSI SQL-92، ولكن قام كل مورد بإضافة مجموعات خصائص خاصة به إلى SQL بغرض تمييز منتجه في السوق ومحاولة عرض أوسع نطاق من الخصائص.

وSQL هي لغة كاملة الخصائص ولكن تختلف كلية عن لغة فجوال بيسك ولغة C. فهي بحق لغة قواعد البيانات العلائقية ولابد أن تكون، حسب ما ورد في البحث الأصلي، الطريقة الوحيدة للتفاعل مع قاعدة البيانات. وفي واقع الأمر، يتم إرفاق SQL غالبا مع التقنيات الأخرى، وخاصة

مع الإجراءات المخزنة. فمع فجوال بيسك وC، قد تأخذ وقت كبير لبيان كيفية تنفيذ كـل خطـوة بطريقة إجرائية، مع تعريف كل خطوة في العملية بالترتيب. ولكن SQL ليست لغـة إجرائيـة، وعادة ما يطلق عليها لغة غير إجرائية أو لغة تصريحيه. بمعنى أنك تقـوم ببيـان مـا تريـده صراحة، وليس كيفية الحصول على هذا الشيء. وقد تأخذ بعض الوقت للتأقلم مع هذا المفهوم إن لم تكن قد تعاملت معه من قبل ولكن الأمر ليس بالصعوبة التي يبدو عليها.

وتتضمن SQL أوامر تقوم بإدخال بيانات جديدة إلى جداول قواعد بيانات وتقوم بتحديـث أو حذف البيانات الموجودة، كما تقوم باختيار بيانات محددة من هذه الجداول. وكذلك تتضمن عـدد كبير من الأوامر الأخرى تستخدم لإنشاء كائنات في قاعدة البيانات والحفاظ عليها وتبديلها: وقـد قمنا بتغطية أغلب هذه الأوامر في الفصول من ٣ إلى ٦. وفي هذا الفصل، سنركز على قابليـة استرجاع البيانات بواسطة إنشاء استعلامات بأمر SELECT.

ويعتبر إتقان إنشاء الاستعلام باستخدام SQL أمرا هاما. فهي تدخل في كل شيء من إعـداد لعنصر تحكم بيانات إلى بعض التقنيات المتطورة التي تستخدم داخل الإجراءات المخزنة والتـي أصبحت متاحة الآن لمطوري فجوال بيسك. سنقوم بتغطية عبارات SQL التـي تقـوم بتحديـث البيانات في الفصل ٨.

أما عن المصطلحات، فقد اعتاد مستخدمو مايكروسوفت واكسل على اعتقاد أن جداول قواعـد البيانات تقوم بتضمين تقارير كثيرة، وكل تقرير يتكون من عدة حقول. ويعتقد مستخدمو أوراكـل أن الجدول الذي يتكون من صفوف يتضمن عدة أعمدة. وفي الحقيقة ليس هناك ثمة اختلف بيـن هذين المصطلحين. ففي هذا الكتاب، سيتم استخدام مصطلحات صف "row" وعمـود "column" في المكان الملائم.

بعض أنواع أوامر SQL

يوجد الكثير من الأنواع المخلفة لأوامر SQL. ففي أوراكل، يوجد ثمانية أنواع، كما هو مفصـل في الجدول ٧-١.

الجدول ٧-١: الأنواع الستة الرئيسية لعبارات SQL

نوع SQL	الوصف
QUERY Command	هذا هو أمر SELECT وهو الطريقة الوحيدة لاسترجاع المعلومـات من قاعدة البيانات داخل لغة SQL.

الجدول ٧-١: الأنواع الستة الرئيسية لعبارات SQL

نوع SQL	الوصف
Data Definition Language (DDL)	تقوم هذه الأوامر بإنشاء أو تبديل أو إسقاط كائنات قاعدة البيانات، بما في ذلك ملفات الجداول والجداول والفهارس، كما تؤدي بعض المهام الإدارية،مثل إدارة المستخدمين.
Data Control Language (DCL)	تستخدم هذه الأوامر لإدارة الامتيازات الممنوحة للمستخدمين.
Data Manipulation Language (DML)	تقوم هذه الأوامر بتغيير البيانات الموجودة في الجداول المنشأة بالفعل (مثل SELECT وINSERT وUPDATE وDELETE).
Transaction Control Language (TCL)	تقوم هذه الأوامر بتثبيت أي تغييرات تطرأ على البيانات في حالة إذا ما تم تغيير تلك البيانات (مثل COMMIT وROLLBACK وSAVEPOINT).
Session Control	تتحكم هذه الأوامر في دورة المستخدم النشطة.
System Control	تقوم هذه الأوامر بتبديل طبعة قاعدة البيانات النشطة.
Embedded SQL	يمكن استخدام هذه الأوامر في لغة إجرائية، مثل اللغات التي وفرها مترجمو أوراكل الأوائل لـ C وPL/I. ويمكن أن تتضمن DDL وDML وTCL وأوامر أخرى. وهذه الأوامر غير متوفرة لفجوال بيسك.

وفي هذا الفصل والفصل التالي، سنناقش أوامر Query وData Manipulation Language. وتم تغطية أوامر Data Definition Language في الفصل ٦ "تنفيذ قاعدة البيانات". وسيتم تغطية عنصر تحكم Transaction في الفصل ٨. وقد تعامل المطورون مع باقي أوامر SQL بشكل عادي، ولكن تفاصيل هذه الأوامر موجودة في مرجع Oracle SQL Reference.

أمر SELECT الأساسي

أسهل طريقة لإدخال SQL، سواء أكانت للأمثلة الموضحة في هذا الفصل أو لأي تجربة تريد القيام بها، هي استخدام دورة SQL*Plus. وبرنامج SQL*Plus متوفر في أي حاسب يدعمه

أوراكل وهو يأتي كجزء من حزمة Oracle Server (خادم أوراكل). على الرغم من ذلك، يمكنك استخدام أي أداة يسهل عليك التعامل معها أو تكون متمرسا عليها أو حتى إذا كنت تفضل استخدامها عن استخدام برنامج SQL*Plus. إذا كنت تستخدم أداة استعلام، تأكد من إنتاجها لـ SQL متوافق مع أوراكل.

صياغة SELECT

الصياغة الكاملة لأمر SELECT متاحة في تخطيط للصياغة (ويطلق عليها التخطيط "المباشر") في مرجع Oracle SQL Reference. وهذا هو عرض مبسط لتخطيط صياغة SELECT:

```
SELECT column_list | *
FROM table_list
WHERE row_conditions join_conditions subquery
ORDER BY sort_columns;
```

وفيما يلي سنقوم باكتشاف عبارة SELECT تدريجيا، وسنقوم كذلك تدريجيا ببناء الاستعلام المركب.

عبارة SELECT البسيطة

سواء كنت تقوم باسترجاع صفوف إلى Recordset Object بسيط باستخدام Data Access Objects التابعة لفجوال بيسك أو بطريقة متطورة ومحكمة بواسطة إجراء مخزن، فأنت دائما تستخدم عبارة SELECT. ورغم احتمال وجود أجزاء أو *عبارات* كثيرة في هذا الأمر، فإن الصياغة الأساسية للأمر تكون كما يلي:

```
SELECT column_list
FROM table_name;
An example of this is
SELECT movie_name, length
  FROM movies;
```

وهذه الصيغة تقوم باسترجاع أعمدة Movie_Name وLength من جدول Movies. إذا كنت تكتب هذا الأمر في دورة SQL*Plus. سيقوم بعرض النتيجة الموجودة على الشاشة بالإضافة إلى عدد الصفوف الموجودة. وجرت العادة على استخدام الحروف الكبيرة لكل حروف SQL الأساسية واستخدام الحروف الصغيرة لأي شيء آخر. يمكن أن تظهر أي عبارة SQL في سطر

أو أكثر وتنتهي دائما بفاصلة منقوطة. ومن المفضل، ولك الخيار في ذلك، بدء كل عبارة في سطر جديد مع تعيين مسافة بدء لها. إذا كنت تقوم بإدخال هذه العبارة في برنامج فجوال بيسك، إذن ستقوم بإدخالها في سطر واحد.

ومكونات أمر SELECT الأساسي هي كما يلي:

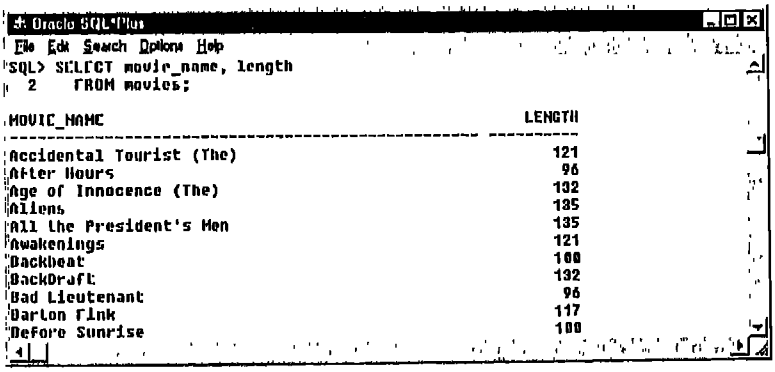
SELECT تقدم تلك الكلمة الأساسية الأمر كاستعلام.

Cloumn list (قائمة العمود) وهي عبارة عن سلسلة من أسماء الأعمدة، وكل منها مفصول عن الأخر بواسطة فاصلة. وسيتم عرض أعمدة مجموعة النتائج بنفس الترتيب إدخالهم في الأمر (على الشاشة أو في Recordset (مجموعة السجلات) إذا كنت قد قمت باسترجاعهم إلى برنامج فجوال بيسك). يمكنك استخدام علامة نجمية بدلا من قائمة العمود لتحديد كل الأعمدة.

FROM تقدم هذه الكلمة الأساسية أسماء الجداول الموجود فيها الأعمدة.

table_name الجدول الذي تريد استرجاع الأعمدة منه.

ونتيجة تشغيل أمر SELECT البسيط موضحة في الشكل ٧-١. سيقوم أمر SELECT باسترجاع كل شيء تريده، ورغم ذلك قد تحصل في بعض الأحيان على أعمدة أكثر من التي طلبتها وربما لا تحصل على شيء على الإطلاق. إذا كنت لا ترغب في عمود بعينه، لا تقوم بطلبه. حتى إذا لم تصل إلى هذا العمود باستخدام Visual Basic Recordset، سيحتاج إلى وقت أكثر بقليل وسيشكل عبئا أكثر على خادم قاعدة البيانات.

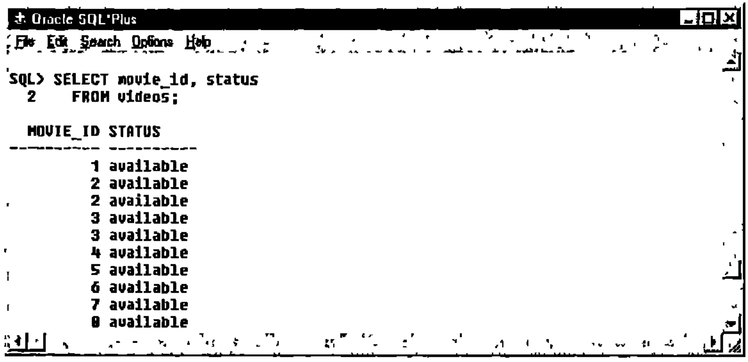


الشكل ٧-١ أمر SELECT البسيط.

إذا كنا قد قمنا باستخدام أمر SELECT هذا في جدول يحتوي على معلومات مكررة، لكنا قمنا باسترجاع تلك البيانات المكررة والمصادر التالفة.

```
SELECT movie_id, status
FROM videos;
```

فهذا الاستعلام، على سبيل المثال، يقوم بإنتاج المخرجات الموضحة في الشكل ٧-٢. فكما ترى، قد ترد كل قيمة لـــ "movie_id" عدة مرات في جدول videos.

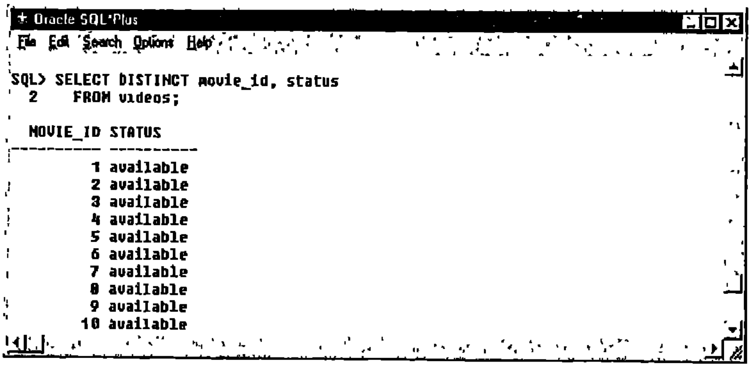


الشكل ٧-٢ استرجاع بيانات مكـــررة بغـــير قصد.

تعطيك SQL القدرة على تقييد الصفوف التي تقوم باسترجاعها بالصفوف المختلفة عن بعضها البض. يمكنك القيام بذلك بواسطة الكلمة الأساسية DISTINCT:

```
SELECT DISTINCT movie_id, status
FROM videos;
```

ونتيجة هذا الاستعلام موضحة في الشكل ٧-٣. إذا قمت بمقارنة هـــذا الشـــكل بالشـــكل ٧-٢، ستلاحظ أننا لا نقوم الآن باسترجاع أي بيانات صفوف مكررة. ومن الملاحظ كذلك أنه بـــالرغم من ظهور "available" مرات كثيرة في عمود status، تنظـــر SQL إلـــى توليفـــة الأعمـــدة Movie_id وStatus عند تحديد صفوف مختلفة.



الشكل ٧-٣ استخدام الكلمـــة الأساســـــية Distinct لاســـــتبعاد الصفـــــــوف المكررة.

وأخيرا، إذا أردت تحديد كل الحقول من الجدول، استخدم علامة نجمية كما في المثال التالي:

```
SELECT * FROM videos;
```

تلميح يأخذ كل حقل تقوم بتحديده موارد خادم وقد يأخذ عرض نطاق ترددي إضافي للشبكة استنادا إلى الأساليب المستخدمة لاسترجاع هذا الحقل. تأكد من أهمية استخدام العلامة النجمية قبل الشروع في استخدامها.

استخدام جدول Dual

يوجد جدول خاص في كل قواعد بيانات أوراكل يطلق عليه dual، وهذا الجدول متاح لكل المستخدمين بطريقة آلية. ويحتوي على عمود VARCHAR2 باسم dummy (عمود وهمي)، وله صف واحد يحمل القيمة X. وتظهر أهمية استخدام هذا الجدول في استرجاع المعلومات من أوراكل وهذه المعلومات غير موجودة في جدول حقيقي. على سبيل المثال، للحصول على التوقيت لخادم أوراكل ، قم باستخدام

```
SELECT SYSDATE FROM dual;
```

وفي هذا الأمر تمثل SYSDATE وظيفة تقوم بإرجاع وقت وتاريخ الخادم. وبهذه الطريقة تكون أقل كمية فقط من تشغيل الجدول هي التي يجب إجراؤها. ويمكنك الحصول على الأرقام المتسلسلة بواسطة استخدام جدول dual (المزدوج)، كما يلي:

```
SELECT my_sequence.NEXTVAL FROM dual;
SELECT my_sequence.CURRVAL FROM dual;
```

تضييق التحديد

ستقوم عبارات SELECT البسيطة، التي تعرفنا عليها حتى الآن، باسترجاع كل الصفوف من الجدول المستخدم، ولكنك ستريد في أغلب الأحيان استخدام تعبير يقوم بتضييق عدد الصفوف.

تحذير تأكد من استرجاع العدد الأدنى فقط من صفوف البيانات من قاعدة البيانات لأداء المهمة المراد تنفيذها. وسيؤدي استرجاع صفوف أكثر من اللازم إلى زيادة التحميل على الخادم وكذلك على الشبكة. بشكل افتراضي، تقوم Visual Basic Data Controls باسترجاع كل الصفوف الموجودة في الجدول، ولهذا يجب أن تكون حريصا عند التعامل معهم.

كما يمكن تقييد الصفوف التي يتم إرجاعها باستخدام عبارة WHERE متبوعة بشرط. والشرط هو تعبير أو أكثر يحتمل إما أن يكون صوابا أو خطأ لكل صف في الجدول المستخدم. ولن يتعجب مبرمجو فجوال بيسك من تلك الشروط، إذ أنها في الأصل عبارة عن جملة IF تم تطبيقها على كل صف بيانات. فعلى سبيل المثال، إذا أردت عرض كل الأفلام (movies) في مدة أقصاها ١٠٠ (مائة) دقيقة، يمكنك كتابة

```
SELECT movie_name
FROM movies
WHERE length<100;
```

يمكنك تجميع التعبيرات في الشرط باستخدام معاملات بولين كما يلي،

```
SELECT movie_name
FROM movies
WHERE length<100 AND rating ='PG';
```

وهكذا فإن آخر حالة تبين أنه في حالة مقارنة عمود حرفي وسلسلة حرفية، يجب أن يتم إحاطة هذه السلسلة بعلامة اقتباس مفردة، وعلى عكس ذلك تستخدم مايكروسوفت أكسس علامة اقتباس مزدوجة. على الرغم من ذلك، فإنه ليس من الممكن القيم بإحاطة القيم الرقمية بعلامات اقتباس.

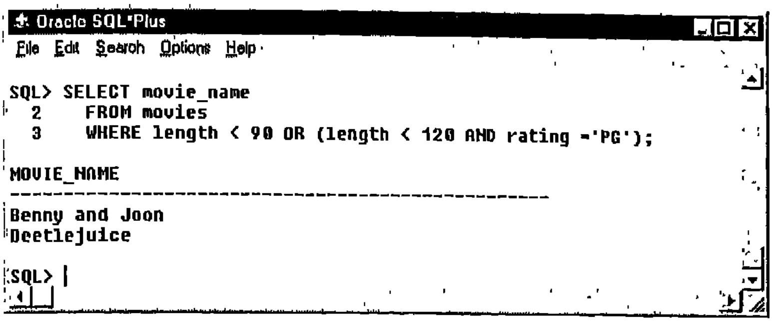
وبالنسبة لمعاملات AND وOR فستكون مألوفة لأعين مطوري فجوال بيسك. وكما هو الحال في كل اللغات، إذا قمت باستخدام معامل OR وكان لديك تعبيرين، فعليك في هذه الحالة استخدام جمل اعتراضية لبيان الطريقة التي تريد أن يعمل الشرط بها، كما في المثال التالي.

```
SELECT movie_name

FROM movies

WHERE length < 90 OR (length < 120 AND rating ='PG');
```

يقوم هذا المثال بالعثور على الأفلام (movies) التي تقل مدة عرضها عن ٩٠ دقيقة وكذلك الأفلام التي تقل مدة عرضها عن ١٢٠ دقيقة ومصنفة PG (المراقبة الأبوية Parental Guidance) (انظر الشكل ٧-٤).



الشكل ٧-[illegible]
استخدام معاملين في عبارة WHERE.

فرز النتائج

وبما أنه يمكنك الآن تحديد صفوف البيانات التي تريدها باستخدام عبارة WHERE، سنقوم بوضع الصفوف في الترتيب الذي نود عرضهم فيه. قد تكون لاحظت أنه حتى الآن لا يمكنك توقع الترتيب الذي ستظهر به الصفوف. وحتى لا يمكنك الجزم بظهورهم في الترتيب الذي تم إدخالهم فيه. ولتعريف الترتيب الذي سيتم استرجاعهم فيه، استخدام عبارة ORDER BY. لا يلزم ظهور الأعمدة المحددة في عبارة ORDER BY في قائمة عمود SELECT.

```
SELECT movie_id, movie_name
FROM movies
WHERE length<100 AND rating ='PG'
ORDER BY length;
```

وكما هو واضح، ستقوم جملة SELECT هذه باسترجاع الأفلام "movies" في ترتيب تصاعدي حسب طولهم. وإذا أردت أن يكون الفرز حسب الترتيب التنازلي، فعليك تتبع اسم العمود باستخدام الكلمة الأساسية DESC كما يلي:

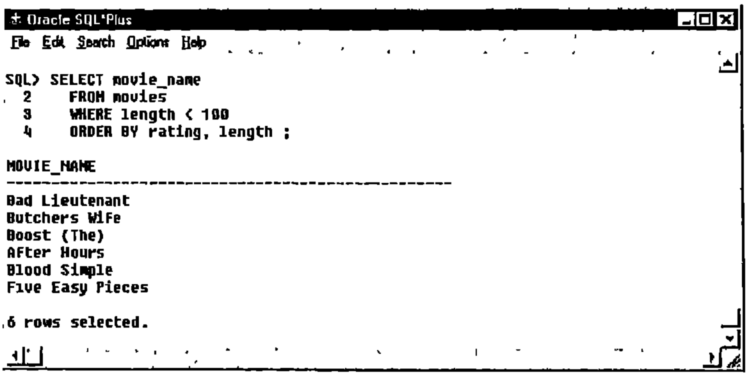
```
SELECT movie_name
FROM movies
WHERE length<100 AND rating ='PG'
ORDER BY length DESC;
```

يمكنك تحديد أكثر من عمود لترتيب الفرز طالما تفصل بين تلك الأعمدة باستخدام الفواصل.

```
SELECT movie_name
  FROM movies
  WHERE length < 100
  ORDER BY rating, length;
```

ستقوم جملة SELECT هذه أولا بفرز الصفوف في مجموعات حسب التصنيف (ويكون أبجديا)، ثم بعد ذلك يتم فرز الصفوف داخل كل مجموعة من مجموعات التصنيف حسب أطوالهم (انظر الشكل ٧-٥).

ملاحظة وبعد التعرف على جملة SELECT، يمكنك فهم الوثائق المرفقة بأوراكل بشكل أحسن. في هذه الحالة، ستحتاج إلى الإطلاع على مرجع Oracle SQL Reference. يقوم هذا المرجع بتخطيط كل هيئة في كل أمر فيما يعرف باسم تخطيط الصياغة أو الطريق المباشر ، وهو عبارة عن سلسلة من الخطوط تحتوي على كل الخيارات المتنوعة. وسنحاول، في هذا الفصل، عرض أوجه الصعوبة لأمر SELECT بالتدريج وليس مرة واحدة.



الشكل ٧-٥ فرز الصفوف بعمودين ORDER BY.

بعض المعاملات المفيدة

لإضافة اللمسات الأخيرة لجملة SELECT الأساسية، يجب علينا الاطلاع على المعاملات التي يمكنك استخدامها في التعبيرات الشرطية في عبارة WHERE. وتعتبر عبارة WHERE هي مصدر التأثير وراء أي سجلات قمت بتحديدها. ويتضمن الشكل ٧-٢ المعاملات الأكثر شيوعا، والتي ستكون على معرفة بها نتيجة لخلفيتك عن برنامج فجوال بيسك.

الجدول ٧-٢: المعاملات الأكثر استخداماً في SQL

المعامل	الوصف	مثال
=	اختبار التساوي	SELECT * FROM movies WHERE rating = 'PG';
!=	اختبار غير التساوي (مثلاً، "غير متساوين")	SELECT * FROM movies WHERE rating <> 'R';
<>		
>	أكبر من	SELECT * FROM movies WHERE length > 90;
<	أصغر من	SELECT * FROM movies WHERE length < 120;
>=	أكبر من أو يساوي	SELECT * FROM movies WHERE length >= 90;
<=	أصغر من أو يساوي	SELECT * FROM movies WHERE length <=120;
||	سلسلة حروف متسلسلة (وهي تشبه & في فجوال بيسك)	SELECT first_name || last_name FROM participants;
AND	صواب إذا كان كلا التعبيرين صواب. وإلا كان خطأ.	SELECT * FROM movies WHERE length <=100 AND rating = 'PG';
OR	صواب إذا كان أي من التعبيرين صواب. وإلا كان خطأ.	SELECT * FROM movies WHERE length > 120 OR rating <> 'R';
NOT	يأخذ العكس المنطقي لنتيجة التعبير.	SELECT * FROM movies WHERE NOT rating = 'PG';

ومن الطبيعي استخدام المعاملات < و> و+> و=< مع البيانات الرقمية، ولكن يمكن كذلك استخدامهم مع مقارنة السلسلة للأعمدة الحرفية. وفي هذه الحالة، تتم مقارنة السلسلة بنفس طريقة التي تتم بها في فجوال بيسك، فعلى سبيل المثال يكون "titanic" أكبر من "monty".

بالإضافة إلى تلك المعاملات القياسية، يوجد بعض معاملات المقارنة الإضافية التي يمكنك استخدامها مع حالة العبارة WHERE وتلك المعاملات متوفرة في معظم قواعد البيانات (انظر الجدول ٧-٣).

الجدول ٧-٣: معاملات إضافية لـ SQL

المعامل	الوصف	مثال
BETWEEN a AND b	يقوم بتحديد الصفوف التي يكون فيها قيمة العمود أكبر من أو مساوية لـ a وأصغر من أو مساوية لـ b	SELECT * FROM movies WHERE length BETWEEN 90 AND 120;
IN	يقوم بتحديد الصفوف التي يكون فيها قيمة العمود مساوية لأي شيء في القائمة	SELECT * FROM movies WHERE rating IN ('PG', 'NC-17');
IS NULL	يقوم بتحديد الصفوف التي يكون فيها قيمة العمود Null (خالية) (غير معرفة)	SELECT id FROM movies WHERE name IS NULL;
LIKE	يتأثر بحجم الحرف، نمط سلسلة حروف تتطابق مع أحرف المثال فيتم استخدام "_" للتطابق التام مع حرف واحد ويتم استخدام "%" لتطابق صفر أو حروف أكثر	SELECT name FROM movies WHERE UPPER(name) LIKE '%JURASSIC%';

استخدام معامل LIKE

يمكنك استخدام معامل LIKE للبحث المستخدم لأحرف المثال، ولكن لابد أن تعلم أنك إذا قمت باستخدام حرف مثال في بداية سلسلة البحث، مثل '%JURASSIC'، يمكن أن يشكل هذا عبئا كبيرا على الخادم، لأنه لن يمكن الاستفادة من أي فهارس. إذا كانت الوظيفة الرئيسية للبرنامج هي إجراء سلسلة بحث، يفضل استخدام خيار أوراكل Context. لا يقوم فقط هذا البرنامج الاختياري بإنتاج فهرس لكل كلمة في المستند، ولكن يقوم كذلك بالسماح بمواضيع البحث الإنشائية والتقريبية. وأصبحت هذه الخاصية المفيدة متوقعة من مواقع ويب. والجانب السلبي يكمن في احتمالية أخذ الفهرس مساحة مشابهة مقارنة بالبيانات الأصلية، ولكن لا يشكل هذا الأمر مشكلة يصعب حلها، إذ أنه يتم توفير موارد كافية كما يمكن حمل الفهرس في قاعدة بيانات أخرى من البيانات.

ويوجد استخدام آخر ممتاز لخيار Context عند التعامل مع تطبيق Yellow Pages يستند إلى ويب، وتعتبر سلاسل البحث السريعة، وهي القدرة على البحث عن أنواع متشابهة من الكلمات، بالإضافة إلى العثور على الكلمات القريبة من بعضها البعض قدرات فائقة وقد تعطي التطبيق ميزة كبيرة.

الأعمدة المحتسبة

قمنا، حتى الآن، في هذا الفصل بعرض قائمة تحتوي على أسماء أعمدة في جملة SELECT يمكن إرجاعها من قاعدة البيانات. ويمكنك بدلا من ذلك تضمين عمود محتسب أو أكثر يقوم بإرجاع قيمة الحساب بين الأعمدة، مثل مجموع عمودين أو قيم نص متسلسلة مع بعضها. ويمكن أن تتضمن الأعمدة المحتسبة معاملات، مثل "+" و"-" و"*" و"/" بالإضافة إلى وظائف أوراكل القياسية.

ملاحظة: لا يتوفر المعامل الدليلي في SQL ولكن يتوفر في PL/SQL على صورة '**'.

تم توضيح أمثلة عن الأعمدة المحتسبة بأسفل. يقوم المثال الأول بإرجاع عمود متعاقب، والأخر يضيف ٨,٥ في المائة لكل الأسعار، ويقوم آخر مثال بإضافة يومين إلى تاريخ التأجير

لإيجاد تاريخ الإرجاع.

```
SELECT first_name || last_name FROM participants;
SELECT cost * 1.085 FROM videos;
SELECT rental_date + 2 FROM videos;
```

معالجة الأعمدة المحتسبة

وجدت في الاختبار الذي أجريته على الأعمدة المُحتسبة في عبارات Select أنه يوجد فرق طفيف في السرعة بين إرجاع حقل متعاقب مرة أخرى إلى فيجوال بيسك وبين إجراء الحساب في فيجوال بيسك نفسه. ولتحديد مكان إجراء الحساب، لابد من ملاحظة بنية التطبيق والموارد النسبية المتاحة للخادم والمزود.

إذا كنت تقوم بتطوير مزود يحتوي على أقل كمية من المعالجة أو إذا كان لديك خادم قوي جداً أو جاري استخدامه، فغالبا ستقوم بالمعالجة في الخادم. إذا كان بنية المزود/الخادم تقليدية، وكانت مناسبة لمعالجة فجوال بيسك لأوراكل أو إذا كان الخادم يفتقر إلى قوة حصان، سيكون من الأفضل إجراء الحسابات في المزود.

وظائف أوراكل المحددة

يحتوي أوراكل على عدد كبير من الوظائف التي تعمل على الأعمدة مثلها مثل أي RDBMS آخر. على الرغم من ذلك، فإن كل قواعد البيانات تطبق مجموعات مختلفة من الوظائف، وحتى الوظائف التي قد تتشابه في الوظائف يمكن أن يقوموا بتطبيقات مختلفة. وفي الجدول التالي تم شرح بعض وظائف أوراكل التي رأينا التأكيد عليها

الجدول ٧-٤: أكثر الوظائف الخاصة بأوراكل استخداما

اسم الوظيفة	الوصف
SYSDATE	تقوم هذه الوظيفة بإرجاع تاريخ ووقت الخادم الحالي.
USER	تقوم بإرجاع اسم المستخدم المنفذ للأمر.

الجدول ٧-٤ : أكثر الوظائف الخاصة بأوراكل استخداماً

اسم الوظيفة	الوصف
TO_CHAR()	تقوم بتحويل البيانات إلى صيغ حروف. وتظهر أهميتها عند التعامل مع التواريخ لأنها تسمح لك بتحديد صيغ مختلفة عن الصيغ الافتراضية، مثل SELECT TO_CHAR(SYSDATE, 'DD-MON-YYYY:HH:MM')، وستقوم هذه الصيغة بإضافة ساعات ودقائق، مثل 21-JUN-1998:12:01.
TO_DATE()	تقوم بتحويل سلسلة إلى تاريخ صحيح إن أمكن، مثل TO_DATE('21 January 1998','DD Month YYYY'). لابد أن تتطابق الصيغة السلسلة الحرفية الجاري تحويلها.
CONCAT	تقوم هذه الوظيفة بتسلسل قيم عمودين فقط، ويعتبر المعامل "||" من المعاملات متعددة الجوانب.
UPPER()	تقوم بإرجاع عمود حرفي في الحروف الكبيرة.
LOWER()	تقوم بإرجاع عمود حرفي في الحروف الصغيرة.
INITCAP()	تقوم بإعداد الحروف أو الكلمات الأولية لتكون في وضع الحروف الكبيرة.

التواريخ في حيز التنفيذ

تعتبر عملية عرض التواريخ ذات أهمية خاصة عند تحديد حقول من جدول قاعدة بيانات. وعادة سيتم استخدام الصيغة الافتراضية، كالتي في المثال التالي:

```
SELECT  SYSDATE  FROM  DUAL;
>  12-SEP-1998
```

إذا أردت عرض عمود تاريخ في صيغة مختلفة، وخاصة للكشف عن معلومات الوقت، لابد من استخدام وظيفة TO_CHAR، وهي تشبه أمر

FORMAT لـ Visual Basic:

```
SELECT          TO_CHAR(SYSDATE,'DD-MON-YYYY:HH:MI:SS')
FROM      DUAL;>12-SEP-1998:13:24:43
```

ستواجه مشكلة بسيطة عندما تريد وضع تاريخ في عبارة WHERE. إذا حددت تاريخ، سيقوم أوراكل بالبحث عن الصفوف التي تحتوي على نفس القيمة في عمود التاريخ. الأمر الذي يعني أن أوراكل سيقوم بمطابقة أقسام التاريخ والوقت لهذا العمود. إذا تم إدخال تاريخ العمود كتاريخ فقط، مثل '12-SEP-98'، سيكون افتراض قسم الوقت 12 A.M. (يتم تخزين الوقت كقسم عشري للعمود ويكون الافتراض Zero (صفر).

إذا أردت البحث عن الصفوف المخزنة بالأوقات والتواريخ، فلابد من توخي الحذر، فعلى سبيل المثال، تحتوي وظيفة SYSDATE على كلا من التاريخ والوقت. فإذا حاولت الآن القيام بـ

```
SELECT * FROM movies WHERE time_stamp = '12-SEP-98';
```

سيصعب العثور على أي صف، لأنك تبحث عن قيم العمود Time_Stamp ووقته 12 A.M (١٢ صباحا). على الرغم من ذلك، يمكنك تفادي ذلك باستخدام وظيفة TRUNC. تقوم هذا الوظيفة باختصار قسم الوقت لعمود التاريخ. ويمكنك الآن القيام بأمر SELECT على النحو التالي:

```
SELECT * FROM movies WHERE TRUNC(time_stamp) = '12-
SEP-98';
```

سنقوم في الفصل ٨ بتغطية طرق وضع صيغ للتواريخ، وفي ذلك الفصل سيكون من الصعب الإلمام الشامل بكيفية معالجة أوراكل للتواريخ.

وظيفة DECODE

تعتبر وظيفة DECODE واحدة من الوظائف التي يصعب فهمها. والذي زاد الأمر سوءا هو تجاهل المراجع لهذه الوظيفة. فقد تم سرد وظيفة DECODE فقط تحت قسم "التعبير الذي تم فك ترميزه" "decoded expression"، ولكن هذه الوظيفة قد تكون أداة ذات قيمة وتشارك في استعلامات أكثر فاعلية.

وأفضل تصوير للوظيفة DECODE هي وصفها بأنها طريقة لتحويل (أو لفك ترميز) تعبير إلى آخر. وقد تتكون هذه التعبيرات من أرقام أو من حروف. كل ما عليك هو إمدادك للتعبير (وغالبا يكون اسم عمود) المراد تحويله وقائمة تحتوي على أزواج بحث/نتائج لتقوم بفك الترميز ثم تتبع ذلك بتوفير قيمة افتراضية ليتم استخدامها إذا لم يكن هناك تطابق بين تعبيرات البحث والنتائج. تقوم صياغة الأمر التالي بتلخيص ما ذكرناه:

```
DECODE (expression, search1, result1,
search2, result2, (etc)
default_value)
```

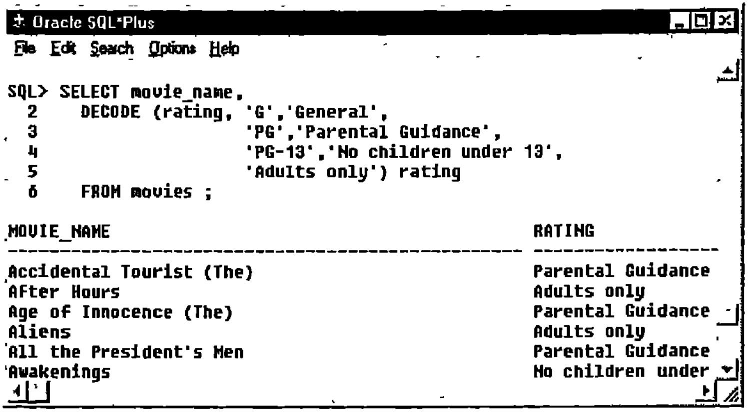
على الرغم من ذلك، فإن المثال التالي سيسهل فهم هذه الوظيفة. فإذا أردت إجراء استعلام عن جدول Movies ولكن تود رؤية نسخة تعليمات برمجة أكثر إيحاءا بدلا من رموز التصنيفات، يمكنك القيام بذلك بواسطة تكوين ترميز جديد وربطه بجدول Movies. على الرغم من ذلك، فيمكنك القيام بذلك بدون جدول ثاني، بواسطة استخدام وظيفة DECODE كما يلي:

```
SELECT movie_name,
 DECODE (rating, 'G','General',
               'PG','Parental Guidance',
               'PG-13','No children under 13',
               'Adults only') rating
 FROM movies;
```

والسؤال الذي يجب طرحه هنا هو ما الذي تقوم به هذه الصياغة؟ عند استرجاع كل صف، يتم في البداية مقارنة عمود التصنيف بـ "G." لو تم التطابق، تقوم جملة SELECT باسترجاع 'General' من ذلك العمود. وإذا لم يتم التطابق، يتم مقارنة عمود التصنيف بـ 'PG' وهكذا. إذا تم العثور على أي صف لا تتطابق فيه قيمة عمود التصنيف مع تعبيرات البحث الثلاث، يتم إرجاع القيمة الافتراضية "Adults Only" لذلك الصف. ويتم توضيح مخرجات جملة SELECT في الشكل ٧-٦.

وتستخدم هذه الطريقة لتحويل القيم الرمزية إلى سلاسل ذات معنى. والعيب الوحيد لهذه الوظيفة هي أنها تعمل بكفاءة أكثر مع عدد قليل من القيم المراد فك ترميزها. ولذلك فإنه من النادر تغيير القائمة، وهو ما يعد شيئا حسنا.

الشكل ٧-٦
استخدام وظيفة DECODE.



```
Oracle SQL*Plus
File Edit Search Options Help

SQL> SELECT movie_name,
  2     DECODE (rating, 'G','General',
  3                     'PG','Parental Guidance',
  4                     'PG-13','No children under 13',
  5                     'Adults only') rating
  6     FROM movies ;

MOVIE_NAME                                         RATING
-------------------------------------------------- ------------------
Accidental Tourist (The)                           Parental Guidance
After Hours                                        Adults only
Age of Innocence (The)                             Parental Guidance
Aliens                                             Adults only
All the President's Men                            Parental Guidance
Awakenings                                         No children under
```

ولجعل عملية إظهار القيم المرجعة سهلة من فجوال بيسك، من الأفضل اتباع وظيفة DECODE وفيها تم استخدام العمود "alias" لإعطاء الوظيفة اسم يمكن استخدامه. والبديل هو سلسلة تتبع اسم العمود وهي مفصولة عنه بمسافة واحدة وتقوم بتوفير أكثر من اسم للعمود في مجموعة النتائج. وفي المثال الذي قمنا بتوضيحه عن وظيفة DECODE، تم استخدام البديل "rating" والذي ظهر كاسم العمود في المخرجات.

وظيفة DECODE في حيز التنفيذ

توضح التعليمات البرمجية ٧-١ مثال يشرح كيفية عمل وظيفة DECODE بطريقة أفضل. وفي تلك الحالة، نحن نتعامل مع قاعدة بيانات تحتوي على جداول مصنفة حسب الساعات، وكل ساعة موضوعة في صف منفصل ويريد المستخدم استعمال وظيفة CROSS-TABULATION الموجودة في مايكروسوفت اكسس ولكن غير موجودة في أوراكل. وتستطيع وظيفة CROSS-TABULATION أخذ البيانات وتكوين ٢٥ عمود، أي عمود لكل ساعة (وتسمح قاعدة البيانات بساعة إضافية لليوم الطويل عندما يزداد عدد ساعات النهار في الخريف) وتسمى "hour1" to "hour25".

ويضاهي التعليمات البرمجية ٧-١ وظيفة CROSS-TABULATION. توجد ٢٥ جملة DECODE، تقوم كل واحدة بتكوين عمود يحتوي على ساعة واحدة في مجموعة النتائج المرجعة، ويتم حساب كل عمود بنفس الطريقة. تختبر كل وظيفة من وظائف DECODE القيمة الموجودة في عمود الساعة للسجل

الحالي. ولو تطابقت مع قيمة الاختبار (رقم ١ على سبيل المثال)، إذا سيتم وضع القيمة الموجودة في عمود Schedule_Amount في عمود الإرجاع، الذي يحتوي على البديل "hour1". إذا لم تتطابق ساعة الاختبار، سيتم إرجاع القيمة الافتراضية صفر.

وبدلا من إرجاع كل الصفوف في الاستعلام، يتم جمع إجمالي كل ساعة بواسطة وظيفة SUM ويتم الجمع حسب Schedule_Day وSchedule_Number، ويتم إرجاع الإجمالي فقط لأنه يتم استخدام عبارة GROUP BY. وبهذه الطريقة يكون مقدار الصفوف التي يتم إرجاعها أقل بكثير مما كانت عليه من قبل.

تعليمات برمجية ٧-١

```
SELECT schedule_day, schedule_number,
   SUM (DECODE (hour, 1, schedule_amount,0)) hour1,
   SUM (DECODE (hour, 2, schedule_amount,0)) hour2,
   SUM (DECODE (hour, 3, schedule_amount,0)) hour3,
   SUM (DECODE (hour, 4, schedule_amount,0)) hour4,
   SUM (DECODE (hour, 5, schedule_amount,0)) hour5,
   SUM (DECODE (hour, 6, schedule_amount,0)) hour6,
   SUM (DECODE (hour, 7, schedule_amount,0)) hour7,
   SUM (DECODE (hour, 8, schedule_amount,0)) hour8,
   SUM (DECODE (hour, 9, schedule_amount,0)) hour9,
   SUM (DECODE (hour, 10, schedule_amount,0)) hour10,
   SUM (DECODE (hour, 11, schedule_amount,0)) hour11,
   SUM (DECODE (hour, 12, schedule_amount,0)) hour12,
   SUM (DECODE (hour, 13, schedule_amount,0)) hour13,
   SUM (DECODE (hour, 14, schedule_amount,0)) hour14,
   SUM (DECODE (hour, 15, schedule_amount,0)) hour15,
   SUM (DECODE (hour, 16, schedule_amount,0)) hour16,
   SUM (DECODE (hour, 17, schedule_amount,0)) hour17,
   SUM (DECODE (hour, 18, schedule_amount,0)) hour18,
   SUM (DECODE (hour, 19, schedule_amount,0)) hour19,
   SUM (DECODE (hour, 20, schedule_amount,0)) hour20,
   SUM (DECODE (hour, 21, schedule_amount,0)) hour21,
   SUM (DECODE (hour, 22, schedule_amount,0)) hour22,
   SUM (DECODE (hour, 23, schedule_amount,0)) hour23,
```

```
  SUM (DECODE (hour, 24, schedule_amount,0)) hour24,
  SUM (DECODE (hour, schedule, sch_amount,0))  hour25
 FROM hours_view
 GROUP BY schedule_day, schedule_number
```

وظائف المجموعة

ورغم أن الوظائف التي رأيناها في الجداول ٧-٢ و٧-٣ تعمل فقط على الصفوف المنفردة، فإن وظائف المجموعة تعمل على مجموعات من الصفوف. وتلك الوظائف تقوم بإرجاع صف واحد فقط من جملة SELECT واحدة. على الرغم من ذلك، ففي المقطع التالي ترى أنه يمكنك تجميع صفوف باستخدام عبارة GROUP BY لإنتاج وظيفة مجموعة في مجموعات متعددة من الصفوف وبالتالي تقوم بإرجاع صف واحد من كل مجموعة. ستكون وظيفة المجموعة الموضحة في الجدول ٧-٥ مألوفة للمبرمجين الذي استخدموا برامج مثل إكسل لأن تلك البرامج تقوم بوظائف مشابهة.

يفضل استخدام الدوال كلما أمكن. ورد الفعل الطبيعي لمطور لفجوال بيسك هو أنك تستطيع القيام بأي حسابات على الحاسب. على الرغم من ذلك، إذا أردت عمل متوسط لعمود في ١٠٠ صف، سيكون عليك جذب ١٠٠ صف من قاعدة البيانات ثم تقم بالحسابات. وعند استخدام دالة AVG، سيكون استرجاعك للعمل أسرع بكثير لأنه يلزم إرجاع قيمة واحدة فقط: المتوسط نفسه.

الجدول ٧-٥: دوال أوراكل التي تعمل على مجموعة من الصفوف

دالة المجموعة	الوصف
AVG (col_name)	تقوم بإرجاع متوسط كل القيم في العمود المحدد.
COUNT (col_name)	تقوم بإرجاع عدد السجلات التي لا تكون قيمة عمودها المحدد خالية. لاحظ حالة (*)COUNT الخاصة، التي تقوم بإرجاع عدد كل الصفوف، بما في ذلك الصفوف الخالية، في القائمة المحددة.
MAX (col_name)	تقوم بإرجاع الحد الأقصى لكل القيم في العمود المحدد.
MIN (col_name)	تقوم بإرجاع الحد الأدنى لكل القيم في العمود المحدد.

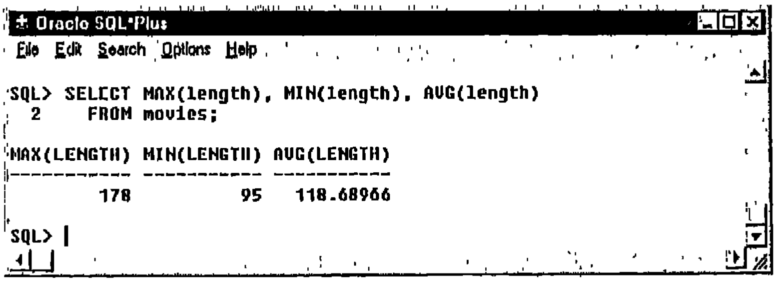
الجدول ٧-٥ : دوال أوراكل التي تعمل على مجموعة من الصفوف

دالة المجموعة	الوصف
STDDEV (col_name)	تقوم بإرجاع الانحراف المعياري الإحصائي لكل القيم في العمود المحدد.
SUM (col_name)	تقوم بإرجاع مجموع كل القيم في العمود المحدد.
VARIANCE (col_name)	تقوم بإرجاع التباين الإحصائي لكل القيم في العمود المحدد.

والطريقة العادية التي ستستخدم بها دوال المجموعة موضحة في المثال التالي:

```
SELECT MAX(length), MIN(length), AVG(length)
  FROM movies;
```

وستعرض جملة SELECT صف واحد يوضح الأحد الأقصى والأدنى ومتوسط طول كل الصفوف المحددة. (يوضح الشكل ٧-٧ مخرجات الاستعلام.) تعتبر هذه الوظائف ذات أهمية ولكن يظهر الجانب السلبي لها عند استخدامها كمدخلات لأوامر SQL أخرى، كما سنرى لاحقا في القسم الخاص بالاستعلامات الفرعية.



الشكل ٧-٧ ثلاث دوال مجموعة في جملة SELECT.

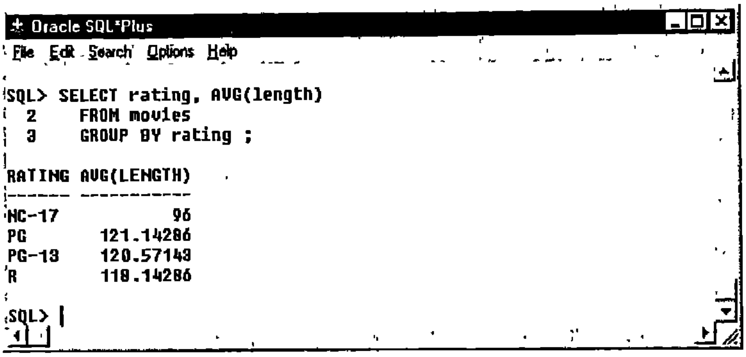
عبارة GROUP BY

في القسم السابق، رأينا كيفية تطبيق وظيفة مجموعة، مثل AVG أو SUM، على مجموعة الصفوف المحددة مرة واحدة. بالإضافة إلى ذلك فإنه يمكنك الفصل بين الصفوف المحددة إلى

مجموعات مميزة ثم تطبيق دوال المجموعة على كل واحدة من المجموعات المميزة. سينتج أمر SELECT خط مخرجات واحد لكل مجموعة. يوضح المثال التالي هذه العملية:

```
SELECT rating, AVG(length)
  FROM movies
  GROUP BY rating;
```

تنتج جملة SELECT تعليمات برمجية لمتوسط طول كل نوع من أنواع التصنيف، كما تقوم بعرض المتوسط بجانب تصنيف الفيلم (movie) كما هو موضح في الشكل ٧-٨. يتم استخدام الأعمدة التي قمت بضمها في عبارات GROUP BY لتعريف المجموعات. عند تغيير أي عمود في هذه العبارة، تبدأ مجموعة جديدة. ففي هذه الحالة، سيشير التغيير في عمود التصنيف rating إلى نهاية المجموعة.



الشكل ٧-٨ عبارة GROUP BY بسيطة.

مشاكل عبارة GROUP BY

هناك أمر لا يتضح مباشرة من المثال وهو أنه بالرغم من إمكانية عرض أكثر من قيمة دالة المجموعة فقط وفي هذه الحالة كانت (طول) AVG لابد من ذكر الأعمدة المرجعة الأخرى في عبارة GROUP BY، الأمر الذي يعني أن تلك الأعمدة تستخدم كذلك لتعريف المجموعات، وهذا ما لا قد تريده، وهذا الأمر قد يكون محبطا في بعض الأحيان، إذ أنك ستحتاج إلى اللجوء إلى أوامر أخرى من SELECT أكثر تعقيدا بكثير أو حتى ستحتاج إلى استرجاع كل البيانات وإنتاج كل الحسابات بنفسك في تطبيق فجوال بيسك.

يمكننا عرض هذه المشكلة في مثال باستخدام دالـــة Max. إذا أردنـــا تحديـــد الحـــد الأقصى لطول الأفلام وكذا أسماء لهم في كل فئة تصنيـــف، لا يمكننـــا كتابـــة

```
SELECT  movie_name,  rating,  MAX(length)
   FROM movies
   GROUP BY rating;
```

في حقيقة الأمر، سنحصل على رسالة خطأ، تشـــير إلـــى عمـــود Movie_Name

```
ORA-00979: not a GROUP BY expression
```

ويرجع ذلك إلى أنه بالرغم من عـــدم العثـــور علـــى التصنيـــف والحـــد الأقصـــى للطول؛ فإنه لا يمكن استرجاع Movie_Name بـــالرغم مـــن عـــدم وجودهـــا فـــي عبارة GROUP BY. إذا أردت معرفة اســـم الفيلـــم الأكـــبر طـــولا فـــي كـــل فئـــة تصنيف، سيكون عليك اســـتخدام أســـلوب آخـــر. ولـــهؤلاء الذيـــن يرغبـــون فـــي معرفة كيفية القيام بهذا في جملة SELECT واحدة، يمكنـــهم التحقـــق فـــي تعليمـــات البرمجة التاليـــة:

```
SELECT  movie_name,  rating,  length
   FROM movies
   WHERE  (rating,  length) IN
      (SELECT rating, MAX(length)
       FROM movies
   GROUP  BY rating);
```

عبارة HAVING

وآخر ما يجب عمله لعبارة GROUP BY هو التحكم في المجموعات المحددة المراد عمل تقريـــر لها. ففي المثال السابق، قمنا بسرد الحد الأقصى لطول الأفلام في كل مجموعات التصنيـــف. إذا أردت فقط رؤية تلك المجموعات التي يزيد فيها طول الفيلم عن ١٢٠ دقيقة، ستحتاج إلى استخدام عبارة جديدة: عبارة HAVING. تعمل هذه العبارة بطريقة مشابهة لعبارة WHERE عـــدا أنـــها تعمل على مجموعات وليس على صفوف فردية. في هذه الحالة، نريد أن نحدد فقط المجموعـــات التي يزيد حدها الأقصى على ١٢٠ دقيقة، لهذا نقوم بكتابة:

```
SELECT rating, MAX(length)
 FROM movies
 GROUP BY rating
  HAVING MAX(length)> 120;
```

ملاحظة حاول تذكر أن عبارة WHERE تقوم تعمل على الصفوف الفردية وأن عبارة HAVING تعمل مع المجموعات. ويمكن أن تظهر كلتا العبارتين في نفس جملة SELECT.

الاستعلامات من أكثر من جدول واحد

إذا كنت قد قمت بإلغاء تطبيع قاعدة البيانات تماما، إذن ستفي الأقسام السابق ذكرها أغلب ما تحتاجه. على الرغم من ذلك، فإنه حتى في قواعد البيانات ذات التصميم السيئ، ستحتاج غالبا إلى استرجاع البيانات من أكثر من جدول. وإذا كنت قد قمت بالتطبيع الكامل لقاعدة البيانات (وهذا ما أتمناه)، فستقوم بتجميع عدة جداول لكل استعلام تقوم بتنفيذه.

ولضم جدولين أو أكثر، ستحتاج إلى القيام بما يسمى ربط الجدول. وهذا هو المصطلح المستخدم للدلالة على ضم أعمدة لأكثر من جدولين. وكما رأينا في الفصل ٥ "تصميم قاعدة البيانات"، أن أساس قواعد البيانات العلائقية هو العلاقات بين الجداول. وتم تعريف هذه العلاقات بواسطة مفاتيح ربط؛ على الرغم من ذلك، فإن مفاتيح الربط ما هي إلا طريقة للتأكد من الحفاظ على تكامل قاعدة البيانات لأن لا تقوم ضمنيا بإنتاج روابط. ولجعل تلك العلاقات ذات فاعلية، لابد من وضع هذه العلاقات في تعليمات البرمجة SQL.

ويوجد أنواع متعددة من الروابط، ويفضل الاهتمام بالصياغة، وبخاصة من له خبرة بتنفيذات أخري لـ SQL، لأنه يوجد بعض الاختلافات "الإقليمية" بين الموردين.

وكما رأينا من أن عبارة FROM في أمر SELECT تسمح لك بتحديد جدول لاسترجاع البيانات منه. في حقيقة الأمر، يمكننا تحديد أي عدد من الجداول إذا قمنا بفصل هذه الجداول باستخدام فواصل. ودعنا نأخذ هذا المثال الذي يعرض لجدولين. فإذا قمنا بكتابة

```
SELECT *

 FROM movies, videos;
```

آملين أن نقرأ كل الأعمدة من كلا الجدولين، وهذا ما سنحصل عليه. على الرغم من ذلك، فإننا سنحصل على أكثر بكثير مما طلبناه في الصياغة. فسيقوم الأمر بتجميع كل صف من جدول مع

كل صف من الجدول الآخر. فإذا كان لدينا ألف سجل في كل صف من الصفين، فإنه سيصبح لدينا الآن مليون صف مجمعين (وهذا ما نطلق عليه الرابطة الديكارتية "نسبة إلى الفيلسوف الفرنسي أرنيه ديكارت" وفي الظروف الطبيعية يجب تجنب هذه الرابطة بأي وسيلة).

ابد من توخي الحذر والدقة عند تحديد ما الذي تريده. فإذا أردت مطابقة صفوف في جدول Videos مع صفوف في جدول Movies حيث تحتوي هذه الصفوف على نفس قيمة movie_id، فستحتاج إلى تحديد هذا باستخدام *شرط ربط* مثل:

```
SELECT *
  FROM movies, videos
  WHERE movies.movie_id = videos.movie_id;
```

تقوم جملة SELECT بإرجاع صفوف من جدول Movies مربوطة بجدول Videos فقط إذا كان هناك أعمدة Movie_id مطابقة في كل جدول. وسيتم تجاهل كل تجميعات الصفوف الأخرى.

قام هذا الاستعلام بعرض مشكلة لم تكن واضحة. ففي المثال السابق، كان لدينا جدولين، وكل منهما يحتوي على عمود باسم Movie_id وهما العمودين المراد ضمهم معا. في عبارة WEHRE، كان علينا تحديد أي الجداول نريد وكان الحل هو تضمين اسم الجدول. فعندما نريد الإشارة إلى عمود في جدول بعينه، نقوم باستهلال العمود بوضع اسم الجدول المحتوي على هذا العمود مفصولا عنه بنقطة، مثل videos.movie_id. وتشبه هذه التقنية تحديد خاصية طبقة في فجوال بيسك حيث كان لابد من فصل اسم الكائن والخاصية باستخدام نقطة.

تكرار أسماء الأعمدة

متى قمنا بإرجاع كل عمود من جدولين أو أكثر، في الوقت الذي يوجد فيه جداول أخرى تحتوي على نفس اسم العمود (كما في حالة Movie_id)، لابد من تفادي تحديد الاثنين. قد تجد أنه لم يتم إرجاع أحد الجدولين، حتى إذا قمت بتحديده لأن محلل أوراكل اللغوي لا يستطيع التفرقة بين الاثنين. قد يكون هذا مربكا عندما تقوم بالاطلاع على Recordset في فجوال بيسك ثم تتعجب من عدم إرجاع أعمدة كافية. ففي المثال، قد تجد أنك تحصل فقط على عمود Movie_id الأول، أي العمود التابع لجدول movies وليس التابع لجدول videos.

قد لا يمثل هذا العمود شيئا بالغ الأهمية (ففي مثالنا كانت قيمة العمودين

واحدة)، ولكن قد يصبح مشكلة فيما بعد. يستحب تحديد الحقول المطلوبة فقط، ولكن إذا أردت كلا الحقلين للعمود، مثل Movie_id، يجب اتباع أحدهم باستخدام بديل للعمود. وهذا اسم بديل للعمود تتم إضافته بعد الاسم الحقيقي للعمود. على سبيل المثال:

```
SELECT   movies.movie_id   movie_id1,
    videos.movie_id  movie_id2
  FROM  movies, videos
   WHERE  movies.movie_id  =  videos.movie_id;
```

تلميح يفضل استخدام مراجع صريحة للجدول ليس فقط لإزالة الغموض ولكن لتحسين الأداء لأن أوراكل يستخدم SQL متكرر لحل مشكلة أسماء الأعمدة الغير معرفة.

تلميح بالإضافة إلى ما سبق ذكره، حاول تجنب ضم العديد من الجداول. فكلما كان عدد الجداول أقل كلما كان أحسن لأن ذلك يعني تحميل أقل على الخادم. وفي كل الأحوال حاول دائما ألا يزيد عدد الجداول عن أربعة.

الروابط الخارجية

يعتبر نوع ربط الجدول الذي رأيناه في القسم السابق ربط معياري في أوراكل، رغم عدم وجود مصطلح معرف بالفعل. وقد يطلق على هذا النوع من الربط في بيئات أخرى ربط داخلي، الأمر الذي يعني تحديد الصفوف من الجدول الأول في حالة التطابق مع صف في الجدول الثاني. وفي المثال، أخذنا كل صف في جدول Movie له تطابق في عمود Movie_id في جدول Video.

على الرغم من ذلك، فقد تريد في بعض الأحيان إرجاع كل الصفوف الموجودة في الجدول الأول بغض النظر عن وجود أي صفوف متطابقة في الجدول الثاني. ويطلق على هذا النوع ربط خارجي. ويمكننا إجراء هذا النوع من الربط في أوراكل باستخدام المعامل (+). نقوم بوضع المعامل في شرط الربط بعد اسم عمود الجدول. فعلى سبيل المثال

```
SELECT *
  FROM movies, videos
  WHERE movies.movie_id = videos.movie.id (+);
```

ستقوم جملة SELECT بتحديد كل الصفوف من جدول Movies سواء كان هناك صفوف متطابقة في جدول Video أم لا. فإذا تم العثور على صفوف متطابقة، إذن سيتم تحديد هذه الصفوف المتطابقة, وإلا، ستكون قيم الأعمدة "المفقودة" خالية "Null". ورغم أن هذا الربط لا يعتبر الربط المعياري، فإنه يوجد حالات كثيرة قد تحتاج فيها إلى استخدام هذا النوع من الربط. على سبيل المثال، يمكننا تضمين أسماء كل الأفلام "movies" في نتائج الاستعلامات حتى إذا لم يكن لدينا نسخة من الفيلم.

ملاحظة ينفذ موردو RDBMS روابط الجدول بطرق مختلفة وباستخدام صياغات مختلفة. لا ينفذ أوراكل صياغة SQL-92 الكاملة للروابط. (ولا يحتوي أوراكل حتى على عبارة JOIN). وفي الناحية الأخرى، وعلى عكس الحال عند أوراكل، تمتلك مايكروسوفت اكسس عدد كبير من الطرق لإنتاج الروابط، بما في ذلك LEFT OUTER JOIN و RIGHT OUTER JOIN . وعلى الأقل فإن صياغة أوراكل أسهل ويمكنها إنتاج أغلب النتائج المطلوبة.

الاستعلامات الفرعية

يعتبر الاستعلام الفرعي تنوع في أمر SELECT، ويؤدي إلى استخراج البيانات بطرق فعالة، على الرغم من أنه في هذه الحالة يصعب فهم الاستعلامات لصعوبتها وكذلك يصعب التحكم فيها. والاستعلام الفرعي هو استعلام يوضع داخل جملة SQL أخرى (غالبا تكون أمر SELECT أو UPDATE أو DELETE يمكن استخدامها مع أمر INSERT) وغالبا يطلق عليه استعلام متداخل.

يستخدم الاستعلام الفرعي لاستخراج معلومات من قاعدة بيانات يمكن استخدامها كمدخلات في استعلام آخر (الاستعلام الأب). في الحقيقة، يمكنك تداخل جمل SELECT أكثر من ذلك، ولكن يجب توخي الحذر، وإلا فلن تحصل على النتائج المطلوبة، وسيصبح الاستعلام مصدرا للإزعاج.

ويوجد نوعين مختلفين من الاستعلامات الفرعية: الاستعلامات التي تقوم بإرجاع صف واحد لا غير، والاستعلامات التي يمكن أن تقوم بإرجاع صفر أو عمود واحد أو أكثر. وتختلف طريقة استخدام النوعين اختلاف طفيف وذلك الاختلاف سيتم شرحه في القسمين الآتيين.

الاستعلامات الفرعية التي تقوم بإرجاع قيمة واحدة

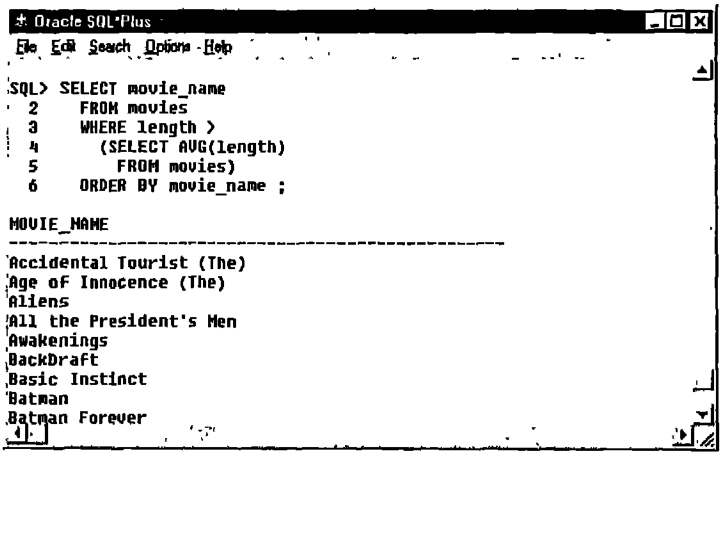
وستقوم غالبا بتكوين استعلام فرعي يقوم بإرجاع قيمة واحدة فقط. ومثال على ذلك هو العثــور على كل الأفلام التي يزيد طول عرضها عن المتوسط. ويمكننا تشغيل اســتعلام للعثــور علــى متوسط الطول ثم نقوم بتركيب استعلام آخر ونقوم بترميز القيمة ترميزا دائما. على الرغم مــن ذلك، فإنه يمكننا عمل هذا مرة واحدة باستخدام الاستعلامات الفرعية:

```
SELECT movie_name
  FROM movies
  WHERE length >
    (SELECT AVG(length)
      FROM movies)
  ORDER BY movie_name;
```

والاستعلام الفرعي موضوع بين جمل اعتراضية ويقوم بإرجاع متوســط طــول كــل الأفــلام الموجودة في جدول Movies كما يلي:

```
SELECT AVG(length)
      FROM movies
```

ويتم بعد ذلك تمرير هذه القيمة إلى الاستعلام الأصلي كقيمة فعلية (مثل ١٠٢) دقيقة. لاحظ أنــه باستخدام هذه الصيغة يجب أن تتم استعادة الاستعلام الفرعي على شكل صف واحــد فقــط. وإلا فسوف تظهر رسالة خطأ.



الشكل ٧-٩ اســـتخدام الاســـتعلام الفرعي للعثــور على الأفلام التي تكون اكبر مــن متوسط الطول.

الاستعلام الفرعي الذي يستعيد أكثر من صف واحد

أحياناً تكون هناك حاجة إلى استخدام الاستعلام الفرعي الذي يستعيد أكثر من صف واحد. ثم يجب استخدام الاستلام الذي يحتوي على الجزء IN من عبارة Where في الاستعلام الأصلي. افترض أنك تريد كل نسخ الفيديو الخاصة بالأفلام الأقل من ٩٠ دقيقة والمتاحة للإيجار، يمكن استخدام التعليمات البرمجية الآتية:

```
SELECT video_id, copy_number
  FROM videos
  WHERE videos.movie_id IN
    (SELECT movie_id
      FROM movies
      WHERE length<90)
    AND status = 'available';
```

ويقوم الاستعلام أولاً بالعثور على قيم Movie_id الخاصة بكل الأفلام التي تكون اقل من ٩٠ دقيقة. وفي هذه الحالة سوف يتم استرداد أكثر من صف واحد. ويتم تمرير القائمة الناتجة من قيم Movie_id إلى الاستعلام الأصلي. ويمكن تحقيق نفس النتيجة باستخدام عبارة SELECT كما يلي:

```
SELECT video_id, copy_number
  FROM videos, movies
  WHERE videos.movie_id = movies.movie_id
    AND videos.status = 'available'
    AND movies.length < 90;
```

وتعتبر طريقة الاستعلام الفرعي أبسط ولكن عامة يفضل تجنب استخدامها نظراً لسوء الأداء. على الرغم من ذلك، فإن في بعض الحالات لا يوجد بديل عملي.

تحسين أداء الاستعلامات الفرعية

من المهم فهم كيفية معالجة أوراكل كجملة SQL. أولا لا بد أن تحلل كل جملة فيما يعرف باسم شجرة الإعراب أو خطة التنفيذ. وتتضمن هذه العملية الخطوات الضرورية لاسترجاع البيانات. وقد وصلت أوراكل لأفضل خطة تنفيذ باستخدام محسن الأداء.

ولأوراكل نوعين من تحسن الأداء. يعتمد النوع الأول على القواعد وهو منتشر منذ فترة. ويوجد حوالي ٢٠ قاعدة "أو مسارات وصول" لتحديد أفضل خطة لجملة SQL معينة. وأسرع

مسار هو Single Row BY Rowid وهو طريق مباشر لصف من البيانات. وتوجد الفهارس في مقدمة القائمة ولكن ليس أعلاها.

ملاحظة A ROWID (معرف الصف) هو موقع أو عنوان لصف واحد في قاعدة بيانات أوراكل ولا يتغير بدأً طيلة مدة سريان الصف. ويعبر عنه باستخدام ROWID pseudocolumn. ويمكنك اختياره من الجدول، ولكن ستراه كجزء من الوصف.

ويعتمد النوع الثاني من محسن الأداء على التكلفة وهو يعمل على التكلفة وهو يعمل على الإحصائيات التي تم تجميعها عن البيانات في الجداول. ولوقت قريب لم يحظ محسن الأداء هذا بسمعة طيبة. الأمر الذي جعل المطورين لا يفضلوا استخدامه. وفي هذا النوع تحدد أوراكل مسارات الوصول باستخدام إحصائيات الجدول التي قامت عبارة ANALYZE بإنتاجها. ويمكنك حساب الإحصائيات بعينها أو حساب الإحصائيات لجدول كبير وكذا تقيمها. ويجب التأكد من تحديث الإحصائيات أولاً بأول.

عبارات SELECT ذات الكفاءة العالية

تعتبر عملية إنشاء قاعدة بيانات عملية رائعة ولكنها تجربة شاقة خاصة لقليلي الخبرة. ومما يزيد الأمر سوءاً إنه ليس ثمة حل سحري فحتى التجربة والخطأ والمعرفة الوثيقة بالتطبيق لا تأتي بطائل. على الرغم من ذلك، يوجد عدد من الأمور العامة التي يجب مراعاتها لتحسين أداء البرنامج.

- قم بوضع أصغر جدول في النهاية. ويرجع ذلك إلى تأثير ترتيب الجداول في عبارة From لجملة select على خطة التنفيذ. على الرغم من ذلك، فإن محسن الأداء المعتمد على التكلفة قد يتجاهل هذه النقطة.
- قم بإنشاء عبارات WHERE بحيث يكون أكثر الشروط تشددا في النهاية ومن الممكن أن يتجاهل محسن الأداء هذه النقطة كذلك.
- حاول تجنب استخدام الوظائف في الاستعلام خاصة في حالة وجود عمود مفهرس، لأن وجود الوظيفة في هذه الحالة سيقوم بتحصيل الفهرس.
- قم بمراجعة WHERE لتحديد إذا كانت مناسبة للفهرس أم لا. إذا كان الجدول صغير أو كانت تقوم جملة Select لاسترجاع كمية كبيرة من البيانات، لن يفيد استخدام الفهرس.

- حاول تجنب وظيفة LIKE المستخدمة لأطراف البدل في البداية على سبيل المثال "movie LIKE '%world'" وقد تكون نتيجة استخدام وظيفة LIKE مسح كامل للجدول عن غير قصد. لا تستخدم الحسابات في عمود مفهرس لأن استخدامها سيعطل استخدام الفهرس.
- يفضل استخدام وظيفة DECODE بدلا من جداول تعليمات البرمجة.
- لا تستخدم الاستعلامات الفرعية غير الضرورية.
- قم بطلب الأعمدة الضرورية فقط. (تقوم "*" بتحديد كل الأعمدة ولكن يكثر مساوئها) إذا قمت بتحديد الأعمدة لتكون العدد المطلوب فقط، يمكنك استخدام طرق مثل Get Rows في فيجوال بيسك لتحسين الأداء.
- قم بطلب الصفوف الضرورية فقط، ولهذا احترس عند التعامل مع عبارات WHERE. وإذا كنت تحتاج إلى تلخيصات استخدام وظائف مجموعة أوراكل مثل AVG وMAX.
- تذكر أنك تريد تجنب قراءات القرص اليدوية كذلك كمية الفرز.

تصميم قاعدة البيانات لجمل SELECT السريعة

ورغم أن موضوع هذا الفصل لا يتصل بتصميم قاعدة البيانات، فإن رأينا إنه من الأفضل إلقاء الضوء على بعض القرارات التي تتخذ عند تصميم قاعدة البيانات والتي ستؤثر على سرعة عمل جمل Select. وبصورة عامة تنقسم قواعد البيانات إما إلى تطبيقات Online Transaction Processing applications، حيث تكون الكتابات، قاعدة البيانات أكثر من القراءات بكثير، أو تكون Decision Support وفيها تكون القراءات أكثر من الكتابات بكثير.

إذا أردت جعل جمل Select في أسرع ما يمكن، سيتوقف هذا على الكتابات. على سبيل المثال، أحد الطرق المفضلة لتحسين أوقات الاستعلام هي استخدام الفهارس على أعمدة متعددة في الجدول. وقد رأينا تصميمات مثل هذه وبدت طيبة للاستعلامات ولكن مالبث أن انتهت في جدول التشغيل عند تحديثها.

ويأتي أصعب القرارات عندما تريد أحسن أداء لكل من القراءات والكتابات ويأتي هذا حل وسط بين أداء القراءة والكتابة والذي يجب عليك تطبيقه بأكثر كمية ممكنة من المعلومات عن التطبيق محل الاستخدام

جملة Explain

بالرغم من أن فهم محسن الأداء مفيدا لأي شخص يريد إنتاج استعلامات ذات كفاءة، وبالرغم من إمكانية استخدام قواعد لتحسين أداء جمل Select، فإنه من الأفضل معرفة أن هناك أداة يمكن استخدامها لاكتشاف أي الطرق حددها محسن الأداء لإنجاز المهمة. هذه الأداة هي أمر SQL، Explain Plan ويمكنها إنتاج ما هو معروف باسم خطة التنفيذ التي استخلصها محسن الأداء.

ولسوء الحظ ن أمر Explain Plan لا يسهل استخدامه كأداة Gui. وهذا الأمر يتكون من ثلاث خطوات وهي كالآتي:

١- تكوين جدول لاستقبال خطة التنفيذ. (وستحتاج إلى عمل هذا في أول مرة تشغيل فقط). تقوم بتكوين الجدول بخط معياري يطلق عليه UTLXPLAN.SQL في معظم الأرصفة #. وفي Windows NT يوجد الملف غالبا في C:\ORANT\RDBMSxx و xx هي رقم إصدار الخادم، مثل ٧٣. بدلا من تكوين هذا الجدول في قاعدة مخطط البيانات بواسطة تنفيذ تنمية الوظائف الفورية من حسابك.

٢- قم بتشغيل أمر Explain Plan. وطريقة تشغيل الأمر موضحة في ملف الوظائف الفورية في التعليمات البرمجية ٧-٢.

٣- قم بتشغيل جملة SQL لترجمة خطة التنفيذ. والوظائف الفورية المعيارية موضح في التعليمات البرمجية ٧-٣.

تعليمات برمجية ٧-٢

```
DELETE FROM PLAN_TABLE;
EXPLAIN PLAN FOR
  SELECT movie_name, video_id
    FROM videos, movies
    WHERE  movies.movie_id = videos.movie_id
    AND videos.status = 'available'
    AND movies.length < 90;
```

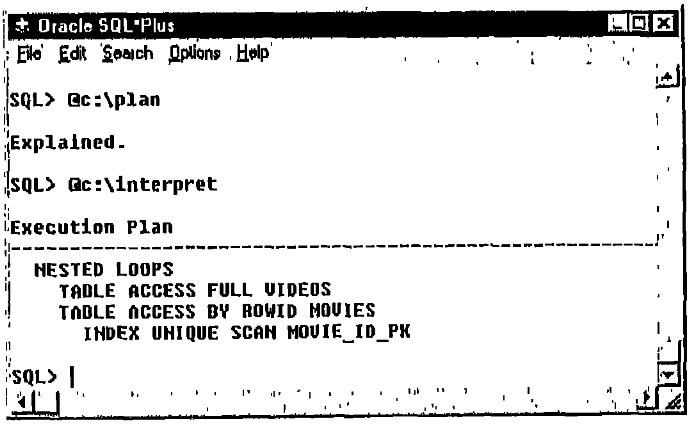
تعليمات برمجية ٧-٣

```
SELECT LPAD (' ',2*level) || operation || ' ' ||
  OPTIONS || ' ' || object_name "Execution Plan"
  FROM plan_table
  CONNECT BY PRIOR id=parent_id
  START WITH id=1;
```

ونتائج تشغيل هذا الخط موضحة في الشكل ٧-١٠. ولسوء الحظ أنه ليسمن الواضح ما الذي تعنيه المخرجات في هذه المرحلة. والمهم معرفة كيفية الوصول إلى الجداول. وهذا الأمر يمكن معرفته من خلال خطوط TABLE ACCESS، والتي لها القيم الممكنة الموضحة في الجدول ٧-٦. كما يوجد معلومات أكثر موجودة في المخرجات. وللحصول على وصف تفصيلي، اطلع على مرجع Oracle Server Tuning.

وفي هذا المثال بالتحديد، يوضح الشكل ٧-١٠ أنه أثناء الوصول إلى جدول Movies بواسطة ROWID (وهو مسح مفهرس)، فإن جدول Videos يدخل تحت مسح كامل للجدول.

الشكل ٧-١٠
خطة تنفيذ لاستعلام بسيط.



الجدول ٧-٦: القيم المتنوعة لخطوط TABLE ACCESS في مخرجات EXPLAIN PLAN

القيمة	الوصف
FULL	تشير إلى مسح كامل للجدول، ولا بد من تجنب هذا إلا إذا كان الجدول صغيرا جدا أو كنت ستقوم باسترجاع معظم الصفوف منه.
CLUSTER	مع هذه القيمة ستكون صلاحية الوصول إلى جدول في تكتل باستخدام فهرس التكتل.
HASH	تشير هذه القيمة أيضا إلى الوصول إلى جدول في تكتل وفي هذه المرة باستخدام فهرس علامة البعثرة.
BY ROWID	تشير إلى أن الصفوف المرجعة تم الوصول إليها بواسطة ROWID. وغالبا تشير إلى إنه قد تم استخدام فهرس.

تحذير تقوم EXPLAIN PLAN بكتابة الصفوف إلى الجدول Plan_Table. ولكن لا تقوم بتنفيذ هذه الصفوف. لا بد من استخدام جملة COMMIT إذا أردت الحفاظ على هذه التغييرات. على الرغم من ذلك، لا بد من حذف كل الصفوف قبل تشغيل EXPLAIN PLAN مرة أخرى، أو ستحصل على خليط من الصفوف لا معنى له.

وتوضح التعليمات البرمجية ٧-٤ جملة EXPLAIN PLAN مأخوذة من استعلام علامة نجمية لأسماء الحقول لأن الاستعلام الفعلي يحتوي على ١٩ عمود معرف. وتوضح المخرجات الموضحة في التعليمات البرمجية ٧-٥ أن الجدولان يتم الوصول إليهما بواسطة المسح الكامل للجدول. وهذا أمر متوقع لأنهما جداول المشتقات صغيرة جدا. والجدولان الباقيان يتم الوصول إليهما بواسطة المفتاح كما تم التصميم.

تعليمات برمجية ٧-٤

```
EXPLAIN PLAN FOR
SELECT  *
  FROM Route_paths, Services, Transactions, Schedules
  WHERE Transactions.Transaction_Date =
    TO_DATE('23-Aug-1998','DD-MON-YYYY')
    AND Schedules.Service_ID = Services.ID
    AND Schedules.ID = Transactions.Schedule_ID
    AND Schedules.Path_Id = Route_Paths.Route_Id
    AND Schedules.type = 'regular'
  ORDER BY path_id, service_id, schedule_id, transactions.id
```

تعليمات برمجية ٧-٥

```
SORT ORDER BY
   HASH JOIN
     TABLE ACCESS FULL ROUTE_PATHS
     HASH JOIN
      TABLE ACCESS FULL SERVICES
      NESTED LOOPS
        TABLE ACCESS BY ROWID TRANSACTIONS
          INDEX RANGE SCAN IND_TRANS_DATE
        TABLE ACCESS BY ROWID SCHEDULES
          INDEX RANGE SCAN SCHEDULES_ID_PK
10 rows selected.
```

تلميح مع كل شاشة فيجوال بيسك تقريبا أقوم بإنشائها لتشغيل الاستعلامات، أفضل وضع مربع نص مخفي يكون مرئيا فقط باسم المستخدم التابع لي فقط. وفي مربع النص هذا أقوم بوضع سلسلة استعلام SQL الحالية. وبجانب كونها أداة تشخيص ممتازة، تسمح لك بالإضافة إلى ذلك بنسخ الاستعلام ولصقه في جملة EXPLAIN PLAN في جملة SQL* PLUS.

تحديد مخطط قاعدة البيانات

ما قمنا بشرحه حتى الآن في هذا الفصل يمكن تطبيقه على أي قاعدة بيانات علائقية. على الرغم من ذلك، عند الوصول إلى بيانات في قاعدة بيانات أوراكل، يجب مراعاة بضعة أشياء مثل مكان وجود بيانات أوراكل، يجب مراعاة بضعة أشياء مثل مكان وجود البيانات وإذا كان لديك صلاحية الوصول إليها أم لا؟

إذا كنت تصل إلى البيانات في مخطط قاعدة البيانات الخاص بك (بمعنى انك أنت صاحب الجدول التي تقوم بالوصول إليها)، إذا أراد مستخدم نهائي (غالبا يقوم باستخدام فيجوال بيسك الخاص بك). الوصول إلى البيانات المخزنة في تطبيقات مخطط قاعدة البيانات، إذا تحتاج إلى تعريف مكان كل جدول في أي من مخططات قاعدة البيانات.

تقوم بتعريف مكان الجدول بواسطة إضافة اسم مخطط قاعدة البيانات على واجهة كل جدول في عبارة From (مفصول، كما هو معروف، بنقطة)، لا تحتاج إلى تحديد مخطط قاعدة البيانات في أي مكان آخر، في عبارة clause مثلا، لأن أوراكل يعلم الآن أي الجداول تقصد. الأمر الذي يعني عدم وجود جداول أخرى تحمل نفس الاسم في مخطط قاعدة بيانات أخر تقوم بالوصول إليه في نفس الاسم في مخطط قاعدة بيانات أخر تقوم بالوصول إليه في نفس الوقت. على سبيل المثال، إذا كانت جداول قاعدة بيانات Movie، موجودة في مخطط قاعدة بيانات Movie، إذن ستكتب لمن لا يستخدم مخطط قاعدة البيانات ما يلي:

```
SELECT *
  FROM video_store.movies, video_store.videos
  WHERE movies.movie_id = videos.movie_id;
```

ويفضل إعطاء الجداول بادئة تطبيق مختصرة، لأنك قد تحتاج إلى استرجاع بيانات من أكثر من مخطط قاعدة بيانات. على سبيل المثال، يفضل تسمية جدول Customers على هيئة Movie_Customers. وإذا كان هناك جدول آخر يعمل معلومات عن العميل في مخطط بيانات آخر، سيكون من السهل التميز بين الجدولين.

إذا كانت الجداول تحمل نفس الاسم، يجب إجراء خطوة إضافية. فكما رأينا مع بدائل العمود، يمكنك الحصول على بدائل جداول كذلك ويكون إضافة اسم بعد اسم الجدول كما يلي:

```
SELECT *
  FROM video_store.movies a, video_store.videos b
  WHERE a.movie_id = b.movie.id;
```

وبالرغم من استخدام الحروف المفردة لا يعتبر البديل المناسب للجداول، فإن استخدامها أصبح تقنية شائعة. ومن المفضل كذلك تبسيط الاستعلامات تقليل كمية الكتابة.

مشاكل الأعمدة الوهمية

رأينا أكثر من طريقة لتحديد صف بيانات أو أكثر من قاعدة البيانات. يقوم أمر Select الرئيسي على عكس الاستعلام الفرعي بتعريف قائمة من أسماء الأعمدة التي سيتم استرجاعها. قد تشير تلك الأسماء إلى الأعمدة الفعلية أو إلى الأعمدة الوهمية، مثل أعمدة NEXTVAL و CURRVAL التي تقوم بإرجاع قيم متسلسلة. يشير مصطلح العمود الوهمي إلى الأعمدة المحسوبة، مثل متوسط القيم في العمود، أو العد، أو تركيب عمودين لإرجاعهم كواحد. ولا يعتبر أي جدول من تلك الجداول حقيقية.

وستتعرف على طرق تكوين والوصول إلى البيانات المرجعة في فيجوال بيسك في الفصول القادمة، ولكن عامة يتم تكوين Recordset بواسطة تمرير جملة Select إلى الخادم. عندما تريد القيمة الفعلية من عمود في Recordset، يمكنك الإشارة إليها باستخدام اسم العمود، وسيعمل طالما كان عمود مناسب. وفي حالة الأعمدة الوهمية، لديك خياران:

- أن تشير إلى العمود بواسطة موضعه في مجموعة Fildes لـ Recordset (تذكر أنه على أساس صفر)
- أن تقوم بإضافة بديل عمود للإشارة إليه. والبديل عبارة عن سلسلة تقوم بإضافتها بعد اسم العمود الموجود بين علامات اقتباس مزدوجة، مثل

```
SELECT first_name || ' ' || last_name full_name FROM customers
```

والبديل full_name تم إرجاعه كاسم العمود.

خلاصة

عرضنا في هذا الفصل أغلب الخصائص التي ستستخدمها عند كتابة جمل Select. كما قدمنا جمل Select الأساسية مع التصفية والترتيب. ورأينا كيفية ربط جدولين أو أكثر معا والتعامل مع الاستعلامات الفرعية. كما قمنا بتغطية بعض النقاط الهامة في أوراكل وفيجوال بيسك. وفي مرجع Oracle SQL Reference ستطلع على خصائص وصياغة أخرى.

ورغم بساطة الأمثلة التي تم تغطيتها، إلا أنها كانت كفيلة لعرض أغلبية هيئات أمر Select. وأثناء تطويرك لجمل أكثر تركيبا، سنقوم بإضافة أسماء أعمدة أكثر لتحديد جداول أكثر وتعبيرات عبارة When أكثر. ولكن ستبقى دائما الأساسيات.

٨

تحديث قاعدة البيانات

بعيداً عن استخدام الاستعلامات لاسترجاع البيانات من قاعدة البيانات، فإن المهمة الأخرى التي ستقوم بها هي كتابة البيانات، فإن المهمة الأخرى التي ستقوم بها هي كتابة البيانات في قاعدة البيانات. فيجب أولاً التعرف على العملية التي تستخدم أوامر Data Manipulation Language SQL، قبل INSERT وUPDATE وDELETE، ويجب التأكد من إجراء العملية بطريقة آمنة وصحيحة وذات كفاءة.

ستقوم في هذا الفصل بتغطية المواضيع التالية:

- فلسفة تحديث الجداول
- إضافة الصفوف
- تحديث الصفوف
- حذف الصفوف
- التحكم في الحركة
- ضمانات أمنية
- زنادات قاعدة البيانات
- تحميل بيانات بكمية كبيرة
- التواريخ ومشكلة عام ٢٠٠٠

فلسفة تحديث الجداول

ما هي أفضل طريقة لتحديث جدول؟ سنراجع في النصف الثاني من هذا الفصل بشكل مفصل الطرق المتعددة التي يتبعها مطور فيجوال بيسك لتحديد بيانات من الجداول وإجراء تغييرات لتلك الجدول. أما الآن فسنلقي نظرة على الفلسفة العامة الكامنة وراء تلك العمليات.

توجد طرق متعددة لتحديث جدول أو العرض من فيجوال بيسك:

١- استخدام Data Control (التحكم في البيانات).

٢- استخدام Recordset (مجموعة سجلات) يمكن تحديثها.

٣- تنفيذ جمل SQL.

٤- تنفيذ إجراء مخزن

وهذه الخطوات مرتبة حسب صعوبتها، ولكنها مرتبة ترتيب عكسي من حيث درجة الأمن والأداء، الأمر الذي يعني أنه يوجد تناوب بين تلك الخصائص.

تجذب عناصر التحكم في البيانات Data Controls أغلب المطورين، خاصة إذا كانوا جدد في برمجة قواعد البيانات. وتأتي مجموعات السجلات التي يمكن تحديثها في المرتبة الثانية. ففي برنامج مايكروسوفت أكسس، لا تعتبر عناصر التحكم في البيانات بديلاً سيئاً. على الرغم من ذلك، بمجرد انتقالك لبيئة مؤسسية، يجب التأكد من استخدام طريقة آمنة وذات كفاءة للاتصال مع قاعدة البيانات. وفي نهاية الأمر يكون تعاملك مع أدوات أوراكل التي تتوحد بشكل كبير مع قاعدة البيانات، كما سيكون عليك إعطاء إمتيازات للمستخدمين بحيث يمكنهم تحديث جداول أوراكل باستخدام مايكروسوفت اكسس أو إكسل أو البرامج التي يكتبها المستخدمون بأنفسهم. الأمر يعتمد على مدى ثقتك بالمستخدمين.

وعلى الأقل يجب أن تكون على دراية بتنفيذ جمل INSERT وUPDATE وDELETE من فيجوال بيسك: يعتقد بعض مسئولي قاعدة البيانات أن هذه الجمل هي الطريقة المناسبة للقيام بذلك. على الرغم من ذلك، فإن جمل DML (لغة معالجة البيانات) لا تقوم بأي حماية، مما يجعل من الصعب وضعها موضع التنفيذ. وتوجد طريقة أكثر أمانا وهي القيام بكل التحديثات من خلال إجراءات مخزنة. وبهذه الطريقة تكفل للمستخدمين امتيازات EXECUTE للإجراءات فقط بدلا من صلاحية الوصول إلى كل جمل DML. إذا لم تكن البيانات ذات قيمة، وسيكتفي بهذا أي مسئول قاعدة.بيانات.

تحذير توجد مشكلة أخرى مع مجموعات السجلات القابلة للتحديث في فيجوال بيسك. فعند استخدامها مع أوراكل، يقوم محلل ODBC بمحاكاة الخلية موضع التحديث في المزود. ونتيجة لذلك، لا يمكنك تضمين أي صياغة مخصصة لأوراكل، مثل الوظائف. ولاستخدام هذه الصياغة لا بد من استخدام خيار SQL Passthrough. ويقوم هذا الخيار بدوره بعدم السماح بتحديث مجموعات السجلات Recordsets وستكون تلك السجلات للقراءة فقط.

سواء قمت بتنفيذ جمل DML من داخل فيجوال بيسك أو قمت بتنفيذ إجراءات مخزنة، ستحتاج إلى فهم كيفية استخدام أوامر INSERT و UPDATE و DELETE، وهذه الأوامر سنقوم بتغطيتها الآن.

إضافة الصفوف

يسمح أمر SQL Insert بإضافة صف أو أكثر إلى الجدول أو الجدول الاعتباري. متى قمت بتكوين سجل جديد، لابد من ملاحظة القيود التي قابلتنا في الفصل ٦، تنفيذ قاعدة البيانات. إذا حاولت تجاهل مطالب القيود، سيفشل أمر Insert.

ويوجد نموذجين لجملة Insert يسمح الأول بإضافة صف واحد فقط، ويستخدم النموذج الثاني استعلام فرعي بحيث يسمح بإضافة صف واحد أو أكثر أو بعد إضافة صفوف.

ملاحظة يمكنك دائما تنفيذ أمر UPDATE في أي جدول في تخطيط قاعدة البيانات الخاصة بك. وإذا أردت استخدام جدول يقع في مخطط قاعدة بيانات آخر، يجب تضمين اسم مخطط قاعدة البيانات، الذي لا بد أن يسبق اسم الجدول مفصول عنه بنقطة، مثل video_store.movies تحتاج إلى امتيازات ملائمة لاستخدام هذا الجدول.

إضافة صف مفرد

يوجد نموذجين أساسين لعبارة Insert للصف المفرد ويستخدم كلا منهم عبارات VALUES لسرد القيم المراد إضافتها في أعمدة الصف الجديد. وتكون صياغة جملة Insert كما يلي:

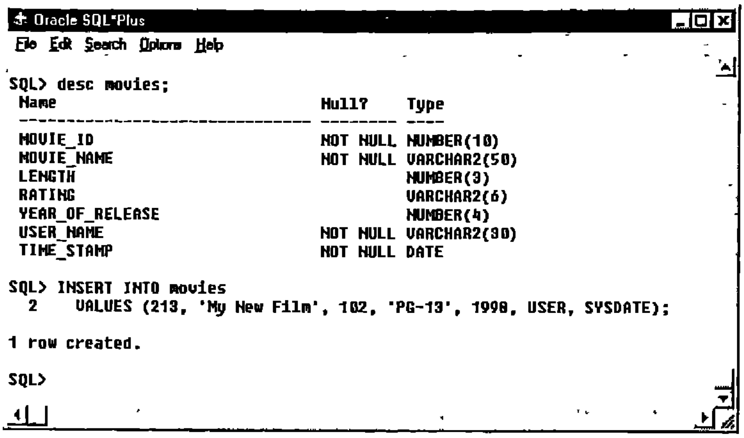
```
INSERT INTO table_name
  VALUES (value1, value2...;
```

وتعتبر العبارة التالية نموذجاً أكثر تفصيلا للصيغة.

```
INSERT INTO movies
  VALUES (213, 'My New Film', 102, 'PG-13', 1998, USER, SYSDATE);
```

عند تنفيذ هذه الجملة (انظر الشكل ٨-١)، ستلاحظ عدد الصفوف التي تتم إضافتها ففي هذا المثال تم إضافة صف واحد فقط. تفترض جملة Insert انك تقوم بإدخال قيمة واحدة في عبارة VALUES لكل عمود في الجدول. يجب أن تتطابق كل قيمة مع كل عمود في نوع البيانات والعدد. إذا لم يتم تتطابق القيم مع الأعمدة، سترى رسالة خطأ إلا إذا كانت القيم الناقصة في نهاية السطر وكانت إما قيمة Nulls وdefault التي قمت بإعدادها في تعريف الجدول.

ملاحظة يحدد أوراكل ترتيب العمود بواسطة الترتيب النسبي لكل عمود في جدول قاموس البيانات الاعتباري DBA_TAB_COLUMNS ويتضح هذا الترتيب النسبي عند تنفيذ أمر DESCRIBE في الجدول.



```
SQL> desc movies;
 Name                            Null?    Type
 ------------------------------- -------- ----
 MOVIE_ID                        NOT NULL NUMBER(10)
 MOVIE_NAME                      NOT NULL VARCHAR2(50)
 LENGTH                                   NUMBER(3)
 RATING                                   VARCHAR2(6)
 YEAR_OF_RELEASE                          NUMBER(4)
 USER_NAME                       NOT NULL VARCHAR2(30)
 TIME_STAMP                      NOT NULL DATE

SQL> INSERT INTO movies
  2    VALUES (213, 'My New Film', 102, 'PG-13', 1998, USER, SYSDATE);

1 row created.

SQL>
```

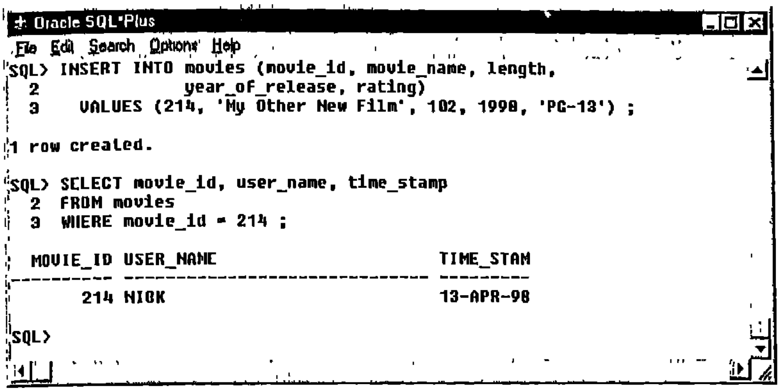
الشكل ٨-١ جملة Insert بسيطة.

يسمح لك النموذج الثاني لجملة Insert الصف المفرد بإدخال فئة جزئية لأعمدة الجدول. ومرة أخرى لابد من مطابقة القيم التي تدخلها مع قائمة العمود. يمكنك تحديد الأعمدة بأي ترتيب تريده طالما تطابق ترتيب القائمة في عبارة VALUES،

```
INSERT INTO movies (movie_id, movie_name, length,
year_of_release, rating)
  VALUES (214, 'My Other New Film', 102, 1998, 'PG-13');
```

وسيتم إعطاء البيانات الناقصة القيمة NULL، إلا إذا تم توفير قيمة DEFAULT في تعريف الجدول ويجب الاستجابة لجميع القيود، فإذا لم تقم بتوفير قيمة لعمود Not NULL وليس له قيمة افتراضية، ستفشل جملة Insert. على سبيل المثال، لا تحتوي جملة Insert في الشكل ٨-٢ اسم مستخدم وختم الوقت الذين تم توفيرهم آليا. وتثبت جملة Select، التي تتبع جملة Insert، أن قيم الأعمدة المفقودة تم توفيرها.

تلميح يمكنك إضافة قيمة NULL في عمود بواسطة تحديد NULL في عبارة VALUES. إذا كان لديك أعمدة يرجح كونها NULL، يجب وضعها في نهاية الجدول. على الرغم من عدم وجود بيانات في عمود NULL، فإن الأعمدة ستأخذ بايت واحد الطول (في هذه الحالة صفر).



الشكل ٨-٢
إمداد قيم الأعمدة آليا مع عبـــارة DEFAULT.

عند إضافة صفوف بيانات، لا بد من التأكد إن كان يتم استخدام مولد متسلسل لتوليد مفـــاتيح فريدة. تم تغطية التسلسلات في الفصل ٦). وبافتراض استخدام المسلســلات، ستســتخدم جملــة Insert أخرى يختلف عن سابقتها اختلاف طفيف:

```
INSERT INTO movies
  VALUES (movie_sequence.NEXTVAL, 'My New Film', 102,
        1998, 'PG', USER, SYSDATE);
```

يتم توليد المفتاح الأساسي الجديد من التسلسل مـــع وظيفـــة Next VAL. إذا أردت معرفـــة الإعداد المولدة، لا بد من استخدام وظيفة التسلسل Curval:

```
SELECT  movie_sequence.CURRVAL from DUAL;
```

إضافة صفوف متعددة

بالرغم من أنك ستقوم في أغلب الأحيان باستخدام أمر Insert لصف مفرد، فإن هنـــاك نمـــوذج آخر لجملة Insert يقوم بإضافة صفوف إلى جدول أو جدول اعتباري تم استرجاعه باســـتخدام استعلام فرعي. وبهذه الطريقة، يمكنك إضافة عدد كبير من الصفوف في جملة واحدة. تفـــترض هذه الخطوة أن البيانات متاحة في جدول آخر. يستخدم هذا النموذج للصيانة لاستخراج البيانات.

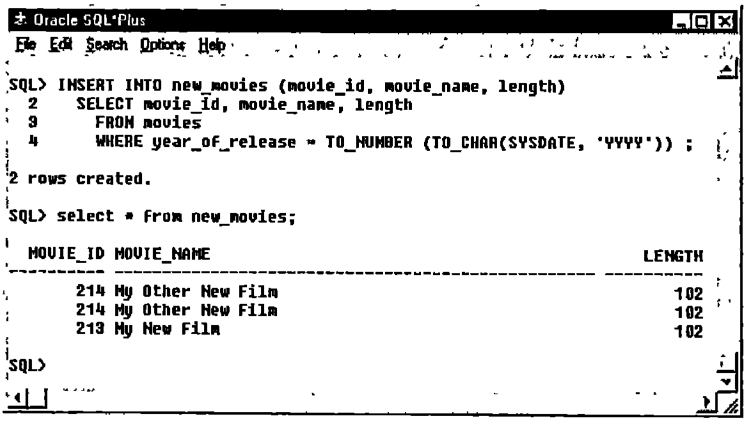
```
INSERT INTO new_movies
  SELECT * FROM movies
    WHERE year_of_release = TO_NUMBER (SYSDATE, 'YYYY');
```

سيقوم الاستعلام الفرعي Select باسترجاع كل الأفلام Movies التي تم إصدارها في العام الحالي وإضافتها في جدول New-Movies. سيعمل هذا الاستعلام الفرعي إذا كان الجدولان Movies وNew_Movies يحتويان على نفس الأعمدة وفي نفس الترتيب. والاسترجاع إلى تحديد قائمة عمود، مثل.

```
INSERT INTO new_movies (movie_id, movie_name, length)
  SELECT movie_id, movie_name, length
    FROM movies
    WHERE year_of_release = TO_NUMBER (TO_CHAR(SYSDATE, 'YYYY'));
```

وتم توضيح هذا المثال في الشكل ٨-٣. يفترض وجود الجدول New_Movies بالفعل. وإذا لم يكن موجوداً، يفضل عدم استخدام أمر CREATE TABLE AS SELECT الموضح لاحقا في هذا الفصل في "تحميل بيانات بكمية كبيرة".

الشكل ٨-٣
جملة Insert تقوم بإضافة أكثر من صف.



```
Oracle SQL*Plus
File Edit Search Options Help
SQL> INSERT INTO new_movies (movie_id, movie_name, length)
  2    SELECT movie_id, movie_name, length
  3      FROM movies
  4      WHERE year_of_release = TO_NUMBER (TO_CHAR(SYSDATE, 'YYYY')) ;

2 rows created.

SQL> select * from new_movies;

 MOVIE_ID MOVIE_NAME                                         LENGTH
--------- -------------------------------------------------- ----------
      214 My Other New Film                                         102
      214 My Other New Film                                         102
      213 My New Film                                               102

SQL>
```

تداول الأعمدة الطويلة

[illegible] Update [illegible] Insert [illegible]

عمود مفرد، ولكن ليس هذا بالحل المرضي وهو بالتأكيد لا يمكن موائمته مع بيانات ثنائية.

وأحد أكبر المشاكل، سواء كنت تستخدم ODBC أو بديل عنه، هي ربط قيمة كبيرة في عمود. ما عليك إلا نقل العمود إلى حزم أصغر، ثم بناء العمود بأكمله في قطع، ثم تجميع هذه القطع مرة أخرى.

وعند التعامل مع فيجوال بيسك، ستحتاج إلى التعود على أساليب AppendChunk وgetChunk. وهذه الأساليب سنناقشها في الفصل ١٣، ولا يوجد بديل لهذه الطريقة إذا أردت تخزين أعمدة كبيرة في قاعدة البيانات.

ورغم إمكانية تصدير هذه الأعمدة الطويلة (حتى ٢ جيجا بيت في أوراكل ٧ و ٤ جيجا بيت في أوراكل ٨)، فإنه يمكن أن يصبح من المستحيل استيرادها مرة أخرى نظراً لقيود الذاكرة وكذلك قيود مقطع إلغاء التعديلات. لا بد من التأكد أن قاعدة البيانات هي المكان الصحيح لتخزين هذه البيانات. يمكن، بدلاً من ذلك، تخزين الملفات على قرص (ربما مضغوط) وتخزين موضع الملف في قاعدة البيانات.

قد يعمل هذا الأسلوب بنجاح إذا كانت الملفات مخزنة في مكان يمكن لمزودي الأجهزة الشخصية الوصول إليه، ربما في خادم ملف بيت وير أو NT. وتؤكد الخبرة أن عند التعامل خلال شبكة، يكون استرجاع الملفات يعمل بكفاءة أكثر من استرجاع أعمدة قاعدة بيانات طويلة.

تحديث الصفوف

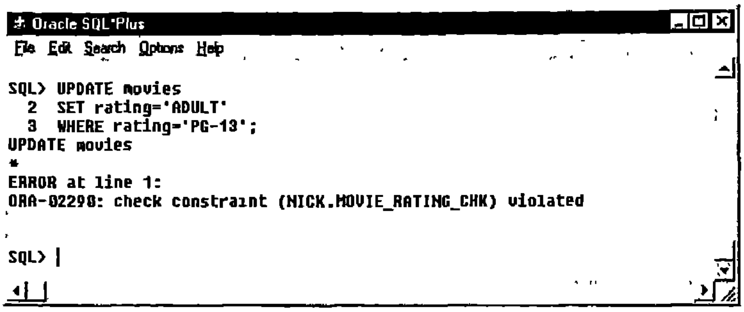
بمجرد إضافة بيانات إلى الجداول، يجب تحديث المعلومات. يسمح لك أمر Update بتبديل عمود أو أكثر لصف بيانات أو أكثر في جدول أو جدول اعتباري. وهذه هي الصياغة.

```
UPDATE table_name
  SET column1 = value1,
      column2 = value2, ....
  WHERE condition_expression;
```

وتحتوي عبارة Set على زوج أو أكثر من أسماء أعمدة وقيم جديدة متطابقة، مفصولة باستخدام فاصلة.

```
UPDATE videos
  SET status = 'available',
     price  = 29.99
  WHERE movie_id = 1243;
```

سيتم تحديث أعمدة Status وPrice بالقيم المعينة للعمود بقيمة 1243 لـ Movie_id. عادة ستستخدم عبارة WHERE لتعريف أي الصفوف يتم تأثيرها، كما هو موضح في المثال. وإذا لم تقم بالتحديد، يتم تحديث كل الصفوف الموجودة في الجدول.



الشكل ٨-٤ جملة Update تتعدى على قيد.

يمكن أن تستند القيمة الجديدة للعمود على القيمة الموجودة لنفس العمود أو لعمود آخر، كما هو موضح في المثال التالي، وفيه زادت الأسعار ٥ بالمائة عند التسليم:

```
UPDATE videos
  SET price  = price * 1.05;
```

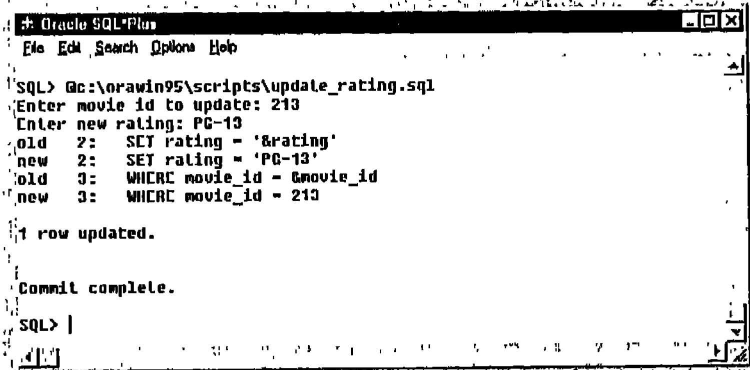
القيم التعويضية في SQL*Plus

إذا وجدت نفسك تكتب سكريبت للحفاظ على البيانات داخل SQL* Plus، لابد من ملاحظة متغيرات التعويض، فهي تسمح لك بتشغيل سكريبت يطالبك بقيم للأعمدة المستخدمة في سكريبت. يتم الإشارة إلى متغير التعويض بواسطة وضع "&" قبل المتغير، على سبيل المثال، &rating. وبشكل افتراضي، سيطالبك SQL* Plus بقيمة للمتغير باستخدام "Enter value for variable:" على الرغم من ذلك، يمكنك استخدام عبارة ACCEPT PROMPT تعريف الموجه

وتوضح التعليمات البرمجية التالية كيفية القيـــام بذلـــك.

```
ACCEPT movie_id PROMPT 'Enter movie id to update: '
ACCEPT RATING PROMPT 'Enter new rating: '
UPDATE movies
  SET rating = '&rating'
  WHERE movie_id = &movie_id;
COMMIT;
```

لابد من تخزين هذه التعليمات البرمجية فـــي ملـــف نـــص، وإلا لـــن تعمـــل بشـــكل صحيـــح. فـــي هـــذا المثـــال، تـــم تخزينـــها فـــي c:\orawin95\scripts\update_rating.sql وتم تنفيذهـــا فـــي المثـــال التـــالي. ستلاحظ حصولك على خطـــوط متعـــددة توضـــح التعويضـــات التـــي قمـــت بـــها. ولإزالة تلك الخطوط، استـــخدم أمـــر SQL* Plus، SET VERIFY OFF.



حذف الصفوف

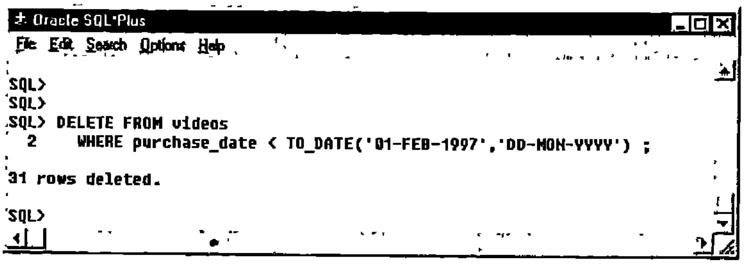
ولآخر جملة في DML هي جملة DELETE، وباستخدامها يمكنك حذف صف أو أكثر من الجـــدول أو نافذة العرض. وتكون الصياغة كما يلي:

```
DELETE FROM table_name
  WHERE condition_expression;
```

ويمكن توضيح هذه الصياغة في التعليمات البرمجية التالية:

```
DELETE FROM videos
   WHERE purchase_date < TO_DATE('01-FEB-1997','DD-MON-YYYY');
```

حيث ستقوم بحذف الأعمدة الموجودة في جـــدول Videos، حيـــث Purchase_Date "تـــاريخ الشراء" قبل فبراير ١٩٩٧ (انظر الشكل ٨-٥).



الشكل ٨-٥ حذف أعمدة من جدول.

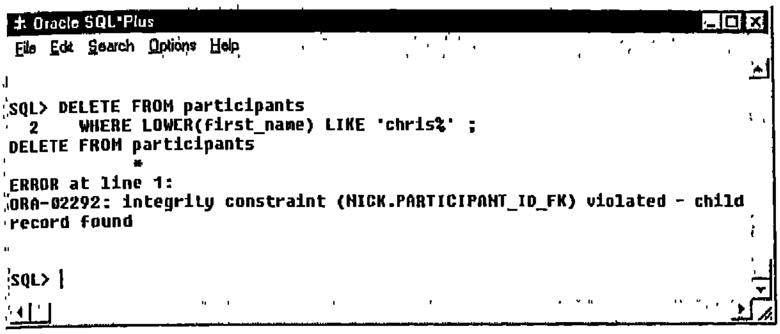
إذا لم تستخدم عبارة WHERE في جملة DELETE، يتم حذف كل الصفوف الموجـــودة فـــي الجدول. ستحتاج إلى استخدام أمر كبديل لذلك:

```
TRUNCATE table_name;
```

وهذا الأمر حرك المؤشر إلى بداية الجدول. وهذا أسلوب ســـريع جـــدا لحـــذف الصفـــوف الموجودة في جدول. على الرغم من ذلك، فإنه لا يمكن إلغاء تعديلات هذا الأمر.

يجب الاحتراس من أن تضم القيود قيد مفتاح ربط. يمكنك إسقاط DROP جـــدول باســـتخدام خيار CASCADE، الذي يقوم بإسقاط كل الجداول المعتمدة عليه من أجل مفاتيح الربط (فهو خيار قوي ولكن خطير في نفس الوقت). على الرغم من ذلك، فليس هناك طريقة سهلة للقيـــام بنفـــس المهمة من خلال أمر DELETE. إذا أردت حذف صف من علاقة أساس تفصيل، ستحتاج إلـــى حذف سجلات التفصيل الملائمة. يعتبر حذف سجل أساسي والسجلات الأبناء هي الطريقة الأمثل للدخول في إجراء مخزن وعمله بواسطة استدعاء إجراء واحد. قم بتمريـــر المفتـــاح الأب إلـــى الإجراء، ثم قم بحذف كل سجلات التفصيل باستخدام نفس المفتاح لحذف السجل الأساسي. وهـــذه الخطوة ستكون محل نقاش في الفصل ١٠، تكوين الإجراءات المخزنة. ويمكن، بدلا من استخدام الطريقة السابقة، استخدام خيار ON DELETE CASCADE عند تعريف قيد مفتـــاح ربـــط فـــي جدول التفصيل. عند حذف الصف الأب، يتم حذفه أي صف أبن كذلك.

إذا حاولت حذف صف مشار إليه في قيد تكامل مرجعي، ستحصل على رسالة خطأ، كما هـــو موضح في الشكل ٨-٦.



الشكل ٨-٦ التعدي على التكامل المرجعي.

إذا أردت إجراء الحذف من نافذة عرض، توجد قيود إضافية لا بد ألا تحتوي نافذة العرض على:

١- شرط ربط

٢- أي معامل مجموعة، مثل GROUP BY أو DISTINCT.

التحكم في الحركة

متى استخدمت جملة Insert أو Update أو Delete في أوراكل، تقوم بإجراء تغيير في قاعدة البيانات، ولكن لا يكون هذا التغيير دائم بشكل افتراضي. وتعتبر هذه التغييرات جزء من حركة، يمكن اعتبارها طريقة لضم تغييرات متعددة، غالبا متصلة، والتي ستنجح أو تفشل كمجموعة. وحتى إذا فشل أحد التغييرات، يجب أن تفشل باقي التغييرات الموجودة في الحركة. عند تثبيت الحركة، تصبح كل التغييرات دائمة في وقت واحد.

فعل سبيل المثال، إذا كانت الحركة في نظام حسابي، فإن عند نقل المال من حساب ووضعه في آخر. فإذا تمت العملية الأولى دون الأخرى، إذن سيتم فقد المال إلى الأبد. وإذا تمت العملية الثانية دون الأولى، يمكنك الحصول على أموال. تقوم الحركة بضم العمليتين بحيث يتم عملهم معا أو لا يتم مطلقا.

وأوراكل عبارة عن نظام يستند إلى الحركة: فبشكل افتراضي، لا يقوم أوراكل بجعل أي تغييرات دائمة في قاعدة البيانات حتى يتم تثبيت الحركة المستحقة. لا يحتاج المستخدم إلى إصدار جملة COMMIT صراحة، فستقوم جملة DDL بإصدار جملة COMMIT ولكن بشكل غير مباشر. على الرغم من ذلك، إذا اطلعنا على كيفية تناول جداول فيجوال بيسك لهذه العملية من خلال ODBC (تبدأ من الفصل ١١، تقنية جانب المزود)، نجد أنه يتم فرض طور Autocommit.

وستحتاج إلى معرفة العديد من أوامر التحكم في الحركة داخل أوراكل، وهذه الأوامر موضحة في الجدول ٨-١.

الجدول ٨-١: أوامر التحكم في الحركة

أمر التحكم في الحركة	الوصف
COMMIT	يقوم بجعل كل تغييرات البيانات المستحقة دائمة.
ROLLBACK	يقوم هذا الآن بتجاهل التغييرات المستحقة ويقوم بإنهاء الحركة.
SAVEPOINT name	يقوم بإعداد علامة مسماة في الحركة الحالية.
ROLLBACK TO SAVEPOINT name	يقوم بتجاهل التغييرات المستحقة لأنه تم إعلان العلامة المسماة.

يبدأ أوراكل الحركة عند تنفيذك أول أمر Insert أو Update أو Delete إما في بداية الجلسة أو بعد نهاية الحركة السابقة مباشرة. وتنتهي الحركة السابقة مباشرة. وتنتهي الحركة عند قيامك بأحد النقاط التالية

- إصدار جملة COMMIT أو ROLLBACK.
- تنفيذ جملة Data Definition Language، مثل جملة CREATE TABLE "أوامر DDL هي نفسها حركات".
- الخروج من دورة SQL* PLUS في تطبيق فيجوال بيسك (وفي هذه الحالة، تم تثبيت الحركة).
- إنهاء دورة SQL* PLUS بطريقة غير طبيعية في تطبيق فيجوال بيسك (في هذه الحالة، تم إلغاء تعديلات الحركة).

ملاحظة يجب مراعاة خاصية التحرك التلقائي لأوراكل. وهذه الخاصية تختلف عن Microsoft SQL Serves، وبالتالي تختلف عن ODBC اللذان يطلبان الصريح للحركة. إذا نسيت تثبيت التغييرات، قد لا تكون المشكلة واضحة لبعض الوقت، ولكن في النهاية سيرى المستخدمون بيانات مختلفة عن التغييرات وبالتالي لن يستطيع تحديث الصفوف المقفلة.

ونظرا لوجود خصائص التزامن لأوراكل، والتي شرحناها في الفصل ٢، بنية قاعدة البيانات لأوراكل، فإن أي تغييرات تقوم بها داخل حركة غير مثبتة لن تكون مرئية لأي مستخدم آخر بخلافك. وتنص خاصية إجراء تغييرات في جدول قاعدة بيانات أوراكل أنه تم قفل أي صفوف بيانات قمت بتبديلها، ولا يمكن لأي مستخدم آخر تغيير الصفوف. ولهذا يفضل حفظ الحركات على أقصر وأسرع وجه. ورغم أن المطورون أصحاب الخبرة الغير كافية مع مخططات قواعد البيانات المقفلة لأوراكل قد لا يعجبوا بذلك، فإن القفل يعتبر مكون حيوي لتنفيذ تكامل البيانات.

التحكم في التحرك في حيز التنفيذ

راجعنا الأوامر المتنوعة التي يمكن استخدامها للتحكم في الحركة، والآن سنطلع على كيفية وضعهم موضع التنفيذ. إذا كنت تستخدم Recordsets يمكن تحديثها أو Data Control في فيجوال بيسك، ستسمح بذلك بالتحديث التلقائي للتغييرات، أو ستستخدم أساليب BeginTrans وCommitTrans وRollback التي قدمتها Data Access Objects للتحكم في الحركة.

وداخل SQL، سواء تم التنفيذ من داخل SQL* PLUS أو إجراء مخزن أو فيجوال بيسك، ستقابل غالبا جملة COMMIT البسيطة مع Rollback في حالة وجود أخطاء. وتم توضيح ذلك الأمر في سكريبت التالي:

```
insert new rows
  update rows
COMMIT;
```

إذا قمت بتنفيذ ذلك من SQL* PLUS، سيتم كتابة أي أخطاء على الشاشة، فسيمكنك تتبع الخطأ باستخدام أمر Rollback. وعند استخدام إجراء مخزن لتغيير البيانات، سيكون من الضروري إضافة أمر Rollback في معالج Excetion إذا قمت باصطياد الخطأ من داخل فيجوال بيسك، سيكون عليك استخدام أسلوب Rollback لإلغاء تعديلات الحركة.

تلميح أفضل طريقة للتعامل مع عنصر الحركة مركب تكون من داخل إجراء مخزن، والذي ستقوم بتوضيحه في الفصل ٩، مقدمة لـ PL/SQL ويوجد بديل للاعتماد على مطور التطبيق، هو استخدام Transaction Server (خادم حركة) لإدارة الحركات.

وتوجد طريقة أكثر صعوبة وأقل استعمالاً في الحركة وهي استخدام Savepoints (نقاط حفظ). ونقاط الحفظ هي طريقة لتعليم مساحة في منتصف حركة طويلة، كما تسمح لك بالرجوع إلى هذه النقطة. ويوضح الاسكريبت التالي هذه الطريقة:

```
insert new rows
  update rows
SAVEPOINT phase1-complete;
  insert new rows
  update rows
SAVEPOINT phase2-complete;
  delete rows
  update rows
ROLLBACK TO phase2_complete;
COMMIT;
```

تم تنفيذ أمر ROLLBACK TO phase2_complete بعد العثور على الخطأ في آخر تحديث للصفوف. هذا يعني أن التغييرات التي تمت للصفوف التي تم حذفها وتحديثها بعد آخر نقطة حفظ تم إلغاء التعديل فيها ويتبع أمر ROLLBACK هذا أمر Commit، الذي يعني أن كل التغييرات قبل نقطة حفظ phase2_complete تم تثبيتها.

الضمانات الأمنية

لتتمكن من استخدام أي جملة من جمل DML هذه في جدول لا يقع في مخطط قاعدة البيانات الخاص بك، ستحتاج إلى الحصول على الامتيازات الملائمة. وقد تم تلخيص هذه الامتيازات في الجدول ٨-٢.

الجدول ٨-٢: الامتيازات المطلوبة لتنفيذ جمل DML

جملة DML	الامتياز المطلوب
INSERT	INSERT ON table_name TO user_name
UPDATE	UPDATE ON table_name TO user_name
DELETE	DELETE ON table_name TO user_name

ولتمكين المستخدم من إضافة صفوف في جدول Movies، استخدم:

```
GRANT INSERT ON movies TO jsmith;
```

ورغم إمكانية إتاحة هذه الامتيازات في كل الجداول للمستخدم، فإنه أكثر أماناً أن تتيح هذه الامتيازات في جدول على حدة. وهذا الأمر لا يشكل صعوبة، لأنك في أغلب الوقت ستستخدم الأدوار (roles)، وقد تم تغطيتها في الفصل ٤ "إدارة قاعدة البيانات"

يمكنك تقييد أعمدة جدول يمكنك إضافتها أو تحديثها بواسطة اتباع الامتياز بقائمة أعمدة في جمل اعتراضية، على سبيل المثال:

```
GRANT UPDATE (movie_name, movie_length) ON movies TO jsmith;
```

تلميح وكما وضحنا خلال الكتاب، إنه من الآمن إعطاء المستخدمين صلاحية Execute فقط، بحيث يمكنك تنفيذ إجراء مخزن، بدلا من إعطائهم امتيازات INSERT وUPDATE وDELETE التي تعطيهم القدرة على تجاهل أي قواعد قمت بوضعها في قاعدة البيانات.

زنادات قاعدة البيانات

متى استخدمت أي جملة من جمل DML التي قمنا بتغطيتها، لا بد من مراعاة إمكانية تنشيط هذه الجملة لزناد قاعدة بيانات. يمكنك تعيين زناد (قطعة من تعليمات البرمجة PL/SQL سنقوم بتغطية هذا الموضوع في الفصلين التاليين) للجدول الذي سيتم تنفيذه عند قيام المستخدم بإجراء Insert وupdate وdelete على الجدول. يمكن تعيين الزناد ليتم تنفيذ قبل أو بعد تنفيذ جملة DML، أو قبل أو بعد تبديل كل صف. يشبه الزناد حدث يتم تنفيذه في عنصر تحكم فيجوال بيسك.

وتبدو الزنادات فكرة رائعة لإعداد مخطط قاعدة بيانات تدقيق، ويمكنها تنفيذ ذلك بواسطة النقل التلقائي للصفوف المحذوفة أو المحدثة إلى جدول تدقيق. كما يمكنها جعل التغييرات متتالية من جدول إلى جداول أخرى في قاعدة البيانات (وربما تبقى على جدول تلخيص يحتوي على أحدث المعلومات). على الرغم من ذلك، يجب الاحتراس عن استخدامها للأسباب التالية:

١- يتم تثبيت إجراءات الزناد سواء تم تثبيت الإجراء العادي أو تم إلغاء تعديله.

٢- كلما زاد عدد الزنادات زادت فرصة تنفيذ هذه الزنادات بعضهم البعض، وكذلك تزيد فرصة تتالى التأثيرات بشكل خطير.

٣- لا يكون تأثير ووجود الزناد واضح لكل مطور.

٤- ويعتبر التدقيق المبني على الزنادات يعتبر تدقيق في أقل مستوى وسينتج ملفات تدقيق ضخمة، والتدقيق في مستوى حركة عالي يعتبر بديل أكثر استخداماً في أحيان كثيرة.

تحميل بيانات بكمية كبيرة

عند تكوين جداول قاعدة البيانات، نحتاج إلى ملئهم بالبيانات. وقد غطينا التقنيات العادية للقيام بذلك مع أوامر SQL. إذا تم إعداد برنامج فيجوال بيسك، يمكننا استخدامه لإدخال البيانات يدوياً. على الرغم من ذلك، قد يكون هناك طريقة أسرع. لا يفضل مع الجداول الجديدة وجود البيانات في جدول آخر أو في قاعدة بيانات يدوياً. على الرغم من ذلك، قد يكون هناك طريقة أسرع. لا يفضل مع الجداول الجديدة وجود البيانات في جدول آخر أو في قاعدة بيانات أخرى، ولكن إن وجدت يمكن استخدام برنامج Import المساعد المذكور في الفصل ٤. على الرغم من ذلك، سنناقش بعض البدائل في المقاطع القليلة القادمة.

تكوين جدول كاستعلام فرعي

أول طريقة لتحميل بيانات ضخمة في جدول تكوين مع آخر SQL، CREATE TABLE AS subquery. وهذا الأمر يعتبر أسهل طريق، ولكنه يفترض وجود البيانات بشكل من الأشكال في قاعدة البيانات. على سبيل المثال.، إذا أردت تكوين جدول يحتوي على البيانات المجمعة من جداول VIdeos وmovIes. يمكننا إجراء أشياء مثل:

```
CREATE TABLE movie_videos
  AS SELECT videos.video_id, videos.movie_id, status, movie_name
    FROM movies, videos
      WHERE videos.movie_id = movies.movie_id;
```

يمكنك إضافة المعاملات الاختيارية العادية لهذا الأمر، مثل تعريف معاملات التخزين وملف الجداول. والسؤال هنا، هل ستقوم عادة بذلك؟ هذا يعتمد على الأمر الذي تريد تحقيقه. فقد يكون الاستعلام مفيدا كذلك لتكوين جداول مؤقتة أو قاعدة بيانات تم إلغاء التطبيع فيها، والتي يعمل على أساسها التقارير.

SQL*Loader

يفترض أسلوب تحميل الجداول أن البيانات لديك بالفعل في جدول، على الرغم من ذلك، فقد ترى عادة أنه يجب نقل بيانات من تطبيق سطح مكتب إلى أوراكل. عندما تريد نقل بيانات عن مصدر

خارجي، ستحتاج إلى استخدام حزم برنامج مساعد يقوم بهذا العمل بنجاح. ورغم وجود بدائل، فإنه يوجد لديك بالفعل في قاعدة البيانات، وهو البرنامج المساعد SQL* LOADER.

وتم بتصميم البرنامج المساعد SQL* LOADER، الذي تقوم بتشغيله من خارج بيئة أوراكل، ليعمل مع نوعين لملفات الإدخال، المعروفين لمطوري فيجوال بيسك. يمكنك تحميل ملف ثابت التنسيق أو محدد بفواصل أو متغير الطول مع إرفاق نص بعلامات اقتباس مزدوجة. يمكن أن تكون هذه العملية، على سبيل المثال، تصدير من مايكروسوفت إكسل أو أكسس أو حتى من جملة Visual Basic Print. ويعتبر أسهل ملف للتعامل معه هو الملف المحدد بفواصل. تقوم بتشغيل برنامج SQL* LOADER من ملف Control، الذي يرشد برنامج SQL* LOADER بتحميل البيانات التي تريدها إلى الجدول الذي تريده، كما يقوم بإمداد معاملات تحكم قليلة. ويوجد مثال يوضح ملف Control في التعليمات البرمجية ٨-١. والصياغة واضحة تماما ولكنها موثقة في مرجع Oracle Utilities لاحظ أن الاسكريبت يشير إلى أن الملفات تنتهي بـ X'09'، وهي حرف جدولة. (تم تصدير البيانات من مايكروسوفت إكسل).

تعليمات برمجية ٨-١

```
LOAD DATA
INFILE *
APPEND
INTO TABLE movies
FIELDS TERMINATED BY X'09'
(movie_id, movie_name, length, year_of_release, rating)
BEGINDATA
1    Accidental Tourist (The)    121    1988    PG
2    After Hours                 96     1985    R
3    Age of Innocence (The)      132    1993    PG
4    Aliens                      135    1986    R
5    All the President's Men     135    1976    PG
6    Awakenings                  121    1990    PG-13
7    Backbeat                    100    1994    R
8    BackDraft                   132    1991    R
9    Bad Lieutenant              96     1993    NC-17
```

لا بد من تشغيل برنامج SQL* LOADER وكذلك تحديد ملف Control. ويختلف الأمر المستخدم لتشغيل ملف Control حسب وجودك على أي حاسب. ولأوراكل ٧-٣ في Windows NT، ستكتب

```
c:> SQLLDR73  control=control_file.ctl
```

يتم مطالبتك بمعرفة المستخدم وكلمة السر. يتم تكوين ملف Log باسم Control_File.log، وسيقوم ذا الملف بتفصيل أي أخطاء تم مواجهتها أثناء التحميل.

وكما في مثال ملف Control، فإن إدخال البيانات يطلق عليها تحميل conventional-path (مسار تقليدي). ويقوم بتكوين جمل INSERT لكل خط بيانات. وهذا الأمر ليس أسرع من كتابة برنامج لتقود بتكوين هذه الجمل بنفسك (إذا كنت تستطيع تشغيل البرنامج على الخادم وتتجنب الأعطال الناتجة عن استخدام الشبكة). ويوجد بديل لذلك، يطلق عليه تحميل direct-path (المسار المباشر)، وهو يقوم بالكتابة مباشرة في الجدول بالتنسيق الداخلي، وهو أسرع بكثير من تحميل المسار التقليدي. على الرغم من ذلك، فأنت تحتاج إلى صلاحية كلمة للجدول، لهذا فاستخدم هذه الطريقة لا تعد مفيدة عن التعامل مع قاعدة بيانات فورية. لا بد من استخدام تحميل المسار المباشر عند تكوين قاعدة بيانات أو عندما يمكنك التشغيل بدون تحديث المستخدمين للجداول أثناء مدة تصميم قاعدة البيانات للتحديث السريع.

تصميم قاعدة بيانات لعمل التحديثات السريعة

رأينا في الفصل ٧، الاستعلامات، بعض التلميحات لتصميم قاعدة بيانات لاسترجاع البيانات في أسرع وقت، وهو الأمر المطلوب في Decision Support System (نظام دعم القرار). على الرغم من ذلك، فإنك تحتاج إلى تحديث سريع في Online Transaction Processing. ولسوء الحظ، هذان النوعان من الأنظمة يتطلبان بيئات مختلفة، فكل ما يقوم بجعل الاستعلامات أسرع، يميل إلى جعل التحديث أبطأ.

وتوجد بعض المفاهيم الهامة يجب استيعابها لجعل التحديث أسرع:

- لا تستخدم فهارس غير ضرورية، فالانتقال من فهرس إلى فهرس يضاعف الوقت لجملة INSERT
- إذا كنت تستخدم تكامل مرجعي بين الجداول، تأكد من تكوين فهرس في العمود المشار إليه. وهذا الأمر يتعارض مع التلميح السابق، لذا يجب التجربة.

- في عبارات WHERE، قم بتحديد المفتاح الأساسي للجدول كلما أمكن.
- تأكد أن الكتابات على القرص تم نشرها بالتساوي في حركات الأقـــراص المتاحـــة. يفضل وجود محركات صغيرة وكبيرة بدلاً من حركات قليلة وكبيرة، بحيث يتم نشــر الكتابات بسمك قليل كلما أمكن.
- اجعل الحركات قصيرة على قدر المستطاع. وإلا، سيتم قفل صفوف البيانــات لفــترة أطول من اللازم. وإذا كنت تحتاج إلى حركات ضخمة، حاول عملهم عندما يكـــون التحميل على النظام قليل، أثناء الليل مثلاً.

تكوين برنامجك الخاص

إذا كان يقوم التحميل التقليدي مع البرنامج المساعد SQL* LOADER بتنفيـــذ جمـــل INSERT فقط، قد يخطر ببالك تكوين برنامجك الخاص للقيام بذلك. وتكوين تطبيق خاص بك لهذا الغـرض به عيب وهو وقت كتابة التطبيق ولكن له ميزات كذلك. وبعيداً عن كون تطبيق كـــهذا مشـــروع عظيم لرفع كفاءات أوراكل، فإنه يعطيك المزيد من التحكم.

التواريخ ومشكلة عام ٢٠٠٠

أصبحت التواريخ نوع بيانات مهم خاصة باقتراب عام ٢٠٠٠. وترجع المشكلة إلى إنه تم تصميم قواعد البيانات للسنوات بحيث يحتوي حقل التاريخ على ستة أرقام باســـتخدام أحـــد التنســـيقات المتنوعة، وكان أكثرها فائدة هي 'yymmdd,' التي تعطي سلسلة تاريخ مثل '980327'، وبـــهذا التنسيق يمكنك على الأقل عمل فرز مرتب زمنياً. على الرغم من ذلك، فإن المشكلة تتمثل فـــي كيفية تمثيل القرن القادم في هذا التنسيق:

فلم يعتقد أي من المصممين أن برامجهم ستصمد كل هذه الفترة بالإضافة إلى الاحتياج إلـــى مساحة التخزين. لا يواجه فيجوال بيسك وأوراكل الآن مشكلة التواريخ، ولكن لا يعني هذا أنـــه يمكنك تكوين تواريخ خاصة بك. فإن كلا المنتجين على دراية بالسنوات الكبيسة عنـــد حســـاب فروق التواريخ، فلن تواجه مشاكل مع هذه النقطة.

تخطي قواعد السنوات

إذا كانت السنة تقبل القسمة على ٤، إذن تكون كبيسة إلا إذا كانت السنة تقبل القسمة على ١٠٠، وإذن لن تكون سنة كبيسة إلا كانت تقبل القسمة على ١٠٠٠، في هذه الحالة تكون سنة كبيسة. إذا عام ٢٠٠٠ سنة كبيسة بينما عام ١٩٠٠ ليس سنة كبيسة.

وتنسيق التاريخ الافتراضي لأوراكل يكون على شكل 'DD-MON-YY, مثل '27-MAR-98,' ؟ ستحصل على 1901 في الوقت الذي تعني فيه 2001. عند إدخال قيمة في هذا التنسيق، يفترض أوراكل أنك تعني هذا القرن، لهذا وإذا أردت إدخال تاريخ في قرن مختلف ستواجه مشكلة.

ويوجد تنسيق بديل لأوراكل وهو DD-MON-RR وتؤدي "RR" وظيفة وهي جعل أوراكل يقوم بترجمة التاريخ كنتيجة ضم التاريخ الحالي والتاريخ الذي تم إدخاله. فيتم الافتراض أن

١- إذا كان التاريخ الحالي في النصف الأول للقرن والسنة المدخلة من صف إلى ٤٩، إذن سيتم افتراض القرن الحالي للسنة التي تم إدخالها. وإذا كانت السنة التي تتم إدخالها من ٥٠ إلى ٩٩، إذن سيتم افتراض القرن السابق لهذه السنة.

٢- إذا كان التاريخ الحالي يقع في النصف الثاني للقرن والسنة المدخلة من صفر إلى ٤٩، إذن سيتم افراض القرن التالي للسنة محل الإدخال. وإذا كانت هذه السنة من ٥٠ إلى ٩٩، إذن سيتم افتراض نفس القرن لتلك السنة.

سارت هذه الطريقة بشكل جيد في بعض الحالات، إلا أننا بلا شك ستسبب إزعاج للمستخدمين عندما تظل إلى السنة ٢٠٥٠. سيكون علينا قراءة القواعد بضعة مرات لاكتشافها. ولهذا السبب لا تعتبر هذه الطريقة الأمثل للاستخدام.

تحـذيـر لاحظ إنه يمكن تغيير تنسيق التاريخ الافتراضي بواسطة معامل نظام، ولكن يجب الاحتراس عند القيام بذلك لأن تعليمات البرمجة التي تستخدمه قد لا يمكن نقله من نموذج أوراكل إلى آخر. ويفضل عدم الاعتماد عليه عند وجودك في أي تنسيق خاص.

وتوجد طرق عدة لتجنب مشكلة تنسيق التاريخ. وأقفل هذه الطرق هي تنسيق سلسلة التاريخ قبل أن يتم إدخالها في قاعدة البيانات وأن تقوم بتنسيقها باستخدام أربعة أحرف للسنة. على سبيل المثال، في فيجوال بيسك، يمكنك السماح للمستخدم بإدخال تاريخ ثم إعادة عرضه مباشرة:

```
Format (my_date, "dd-mmm-yyyy")
```

ولتحرير هذه الصياغة إلى قاعدة البيانات في جملة DML، مثل جملة INSERT، تحتاج إلى استخدام وظيفة أوراكل TO_DATE.

```
TO_DATE (date_string, 'DD-MON-YYYY')
```

وتأخذ وظيفة TO_DATE المعامل الأول (date_string) وتقوم بتحويل هذا المعامل إلى تاريخ التنسيق الداخلي لأوراكل، مفترضا أنه في التنسيق المحدود بواسطة المعامل الثاني ('DD-MON-YYYY'). وهذا التنسيق قد يسبب بعض القلق أحيانا، ولهذا توضح التعليمات البرمجية ٨-٢ وظيفة فيجوال بيسك تحقق إلى ذلك.

تعليمات برمجة ٨-٢

```
Public Function OracleDate(inDate as Date) As String
' Return the date in the form suitable for Oracle stored procedures
  OracleDate = "TO_DATE('" & Format(inDate, "dd-mmm-yyyy") & _
          "','DD-MON-YYYY')"
End Function
```

ملاحظة: لتحديد سلسلة شهر تتكون من ثلاثة حروف، استخدم "MON"، ويستخدم فيجوال بيسك "MMM".

يقوم عمود التاريخ كذلك بتخزين الوقت في ساعات ودقائق وثواني. (في الحقيقة، يتم تخزينها كالجزء العشري لقيمة التاريخ). إذا قمت فقط بإدخال تاريخ، يكون افتراض الوقت ١٢ صباحا. ولإدخال تاريخ ووقت في قاعدة البيانات الخاصة بك، ستحتاج إلى استخدام تنسيق محدود.

```
TO_DATE (date_string, 'DD-MON-YYYY:HH24:MI:SS')
```

يصعب عمل ذلك في فيجوال بيسك، لذلك توضح التعليمات البرمجية ٨-٣ إجـراء فيجـوال بسك سيقوم لك بالعمل.

تعليمات برمجة ٨-٣

```
Public Function OracleDateTime(inDateTime as Date) As String

' Return the date and time in the form suitable
' for Oracle stored procedures
  OracleDateTime = "TO_DATE('" & _
    Format(inDateTime, "dd-mmm-yyyy:hh:mm:ss") & _
    "','DD-MON-YYYY:HH24:MI:SS')"
End Function
```

• **تلميح** عند إدخال عمود تاريخ لختم الوقت، يمكنك استخدام تاريخ النظام مع وظيفة SYSDATE.

خلاصة

قمنا في هذا الفصل بتغطية طرق إدخال وتحديث وحذف صفوف البيانات. تم توفير هذه التقنيات بواسطة جمل DML المتاحة في SQL وهي معيارية في الأنظمة المختلفة لـ RDBMS. كما قمنا بتغطية كيفية فهم وتداول الحركات في أوراكل، ويعتبر هذا الأمر مهما لأنه يختلف عن الأنظمة الأخرى.

وقمنا، في الجزء الخاص بتحميل البيانات بكمية كبيرة، بتغطية كل شيء تحتاجه للتطوير. وستساعدك معلوماتك الآن عن الفلسفة وراء تحديث البيانات على استخدام تقنية آمنة وذات كفاءة.

٩

مقدمة عن PL/SQL

ظهرت قواعد البيانات أساساً لغرض واحد: تخزين البيانات. ولا تعد هـــذه فكـرة غريبـــة لأي شخص نما مع منتجات مثل مايكروسوفت أكسس أو فوكس برو. حتى في منتج مثـــل أكســس، أدرك المصممون أن تخزين الوحدات ليست مجرد أصناف بيانات تقوم بتطوير وظيفـــة المنتـــج بقدر كبير. ففي قاعدة بيانات أكسس، على سبيل المثال، من الممكن تخزين ليس فقـــط الجـــداول ولكن الاستعلامات والنماذج والتقارير ومنظومات الرموز أيضاً. والبديل لذلـــك هـــو توزيعـــهم وتخزينهم محلياً لكل المستخدمين المهتمين بذلك.

ومع المنتجات الموجهة للمؤسسات مثل أوراكل، أسهمت إضافـــة خـــواص ممـــيزة لتحقـــق متطلبات المؤسسات مع رفع مستوى المنافسة. وأكثر خاصية تمت إضافتها عموماً هـــي القـــدرة على تنفيذ رموز البرامج على الخادم لمعالجة البيانات قبل قراءتها أو حفظها. تسمى هذه الخاصية الإجراء المخزن وهذا هو موضوع هذا الفصل والفصل التالي.

والموضوعات التي سوف نقوم بتغطيتها في هذا الفصل هي:

- الإجراءات المخزنة
- الحزم
- أدوات تكوين الإجراءات، وتتضمن SQL*PLUS
  - SQL*PLUS
  - أدوات قاعدة البيانات المرئية
  - باني الإجراءات لأوراكل
- معالجة الأخطاء في PL/SQL.

PL/SQL كامتداد لـــ SQL

"SQL" Structured Query Language أو(لغة الاستعلام الهيكلية) عبـــارة عـــن مجموعـــة الأوامر التي تمكنك من معالجة البيانات قاعدة البيانات العلاقية. وSQL لغة غير إجرائية (وتسمى غالبا لغة بيانية) التي تعني بيان النتيجة التي تريدها بدل من شرح كيفية إجرائها بالضبط. علـــى سبيل المثال، لا توجد حلقات أو تركيبات IF-THEN-ELSE في SQL.

بالرغم من أن SQL قوية وسهلة الاستخدام، إلا أنها تشبه برمجة الفيجوال بيسك عند استخدام الفأرة. يمكنك عمل نماذج جذابة (وحتى تكوين بعض نظم إدخال البيانات الابتدائية بالتحكم فـــي البيانات)، ولكن لعمل تطبيق مفيد حقيقة بالفيجوال بيسك، عليك إضافة تعليمات البرمجة لتحديد ما تريد وكيفية تنفيذه. SQL نفس الشيء وكل مورد قاعدة بيانات علاقية يأتي بامتداد SQL الخـــاص به. وبالنسبة لأوراكل، يسمى الامتداد PL/SQL، تعليمات برمجة PL إلى اللغة الإجرائية.

تحتوي منتجات أوراكل المتعددة على محركات PL/SQL التي تقوم بتنفيذ نفس الـــدور مثـــل مفسر مشغل الفيجوال بيسك. ولسوء الحظ، حتى بالرغم من أن PL/SQL قائمة على أساس لغـــة البرمجة ADA، فإنها لا تشبه لغات مثل البيسك إلى حد بعيـــد ويتـــم تنفيذهـــا ببـــطء باســـتخدام جافا.ولكن الفيجوال بيسك توجد في مكان العمل الحالي وفي المكان الذي يحتاج إلى بذل طاقـــة. ومن الناحية الإيجابية، لا توجد تركيبات متعددة جداً لتعلمها.

الإجراءات المخزنة غير مرتبطة بأي وحدات قاعدة بيانات مثل الجـــداول أو المسلســـلات. فالإجراءات مرتبطة فقط بمخطط قاعدة البيانات الذي تم فيه تكوينهم. ومع هذا يوجد اعتماد قليـــل على هذه الوحدات الأخرى. على سبيل المثال، إذا كان الإجراء المخزن يشير إلى جدول يتم على التوالي، فسوف يتم تعليم الإجراء المخزن كتجميع غير صحيح على الرغم أنك تكـــون لا تـــزال قادر على تنفيذه. وفي حالة جملة DELETE، قد يعمل لإجراء المخزن بينما قـــد تفشـــل جملـــة INSERT في حالة تغيير عدد أو نوع الأعمدة. وسوف نتعرف على كيفية عمل هذه العملية فـــي الأقسام القليلة القادمة.

الإجراءات المخزنة

الإجراء المخزن عبارة عن قطعة من رموز يتم تخزينها في قاعدة البيانات وتقوم بتنفيذ وظـــائف متنوعة عادة ما تستخدم بيانات القراءة من أو بيانات الكتابة إلى قاعدة البيانات. يمكن اســـتدعاء الإجراءات بواسطة تطبيقات خارجية عن قاعدة البيانات (برامج فيجوال بيسك، على سبيل المثال) كما أنها من الممكن للإجراءات السرعة للبيانات للمرور للأمام والخلف بينهم وبين التطبيق فـــي شكل المعاملات. يتم تكوين وإدخال الإجراءات المخزنة داخل مفسر قاعدة البيانات فـــي تنسـيق داخلي يسمى P. Code.

عادة ما تستخدم مصطلح الإجراء المخزن بشكل عام مع أنه يفضل تطابقه على الإجـــراءات الذي يعني نفس مفهوم النظام الفرعي للفيجوال بيسك. يستخدم أوراكل مصطلح البرنامج الفرعـــي الذي يشير ليس فقط إلى الإجراءات ولكن أيضا إلى الوظائف والزنادات. فقد تكـــون الوظـــائف قامت باستخدام نوع البرنامج الفرعي الذي يقوم بتنفيذ نفس الدور كالوظيفة في الفيجوال بيســـك. والزنادات عبارة عن برامج فرعية يتم ظهورها حرفيا لتنفيذها عند إدخال وتحديث وحذف صـــف في الجدول مثل الحدث.

يتم تخزين الوظائف والإجراءات المخزنة في مخطط قاعدة البيانات للمستخدم ومالك المسئول عن قاعدة البيانات سوف يقوم بالترتيب لمستخدمين آخرين ليكون لهم السماح في تنفيذهم إذا كـــان ذلك طلب التطبيق. وأفضل طريقة لإعداد الإجراءات المخزنة هي تكوين مخطط قاعدة بيانـــات للتطبيق (بمعنى مخطط قاعدة البيانات الذي لا ينتمي إلى أي مستخدم حقيقي) الذي يتم فيه تخزين

كل الإجراءات لذلك التطبيق. ولن تكون بحاجة لامتيازات بالنسبة للزنادات، ففي الحقيقة لا يمكنك وقف ظهور الزنادات الخاصة بهم لأن ذلك سوف يحدث آلياً ما لم يتم التعطيل.

مميزات الإجراءات المخزنة

تحتوي البرامج الفرعية المخزنة على مميزات متعددة لمصمم التطبيق وقاعدة البيانات. وعلى الرغم أن البرامج الفرعية المخزنة تكون عادة أقل شيء أهمية بالنسبة لمبرمج الفيجوال بيسك، إلا إنهم الطريقة الوحيدة التي يمكنك من خلالها التعامل مع البيانات.

ومميزات الإجراءات المخزنة كالآتي:

دفق بيانات منخفض: تنفيذ تعليمات البرمجة على خادم قاعدة البيانات يعني عدم وجود ازدياد مرور في الشبكة منفصل عن إرسال معاملات قليلة للأمام والخلف. وفي النظام على أسس الملف مثل أكسس تتطلب بعض العمليات برنامج الوسيط لسحب كل السجلات للبحث عن القليل فقط. قد يؤدي هذا إلى الأنظمة التي تكون غير قابلة للتدرج مطلقا.

دفق رموز منخفض: إذا كان يوجد لديك قطعة معقدة من الرموز، فسوف تكون موارد الشبكة عاجزة عن تحويلها إلى قاعدة البيانات في كل مرة تريد تنفيذها. ويخزنها في قاعدة البيانات يتم حل المشكلة.

أداء متطور: تنفيذ تعليمات البرمجة في قاعدة البيانات يعني أنه سوف يقوم بالتنفيذ بأسرع شكل ممكن. ومع أن تعليمات البرمجة لا تمر في معنى كلمة فيجوال بيسك، إلا إنه يتم تحليل خطة للتنفيذ تكون في وضع تنفيذ آخر ويجب إجراءها عند وقت التشغيل.

وضع قوانين العمل: بإعطاء المستخدمين الامتيازات لتنفيذ الإجراء المخزن، يمكنهم معالجة البيانات مباشرة من خلال الإجراء، ومع هذا يمكنك التأكد أنه لا يوجد لهم أي امتيازات لجداول الأساس ذاتها. وعلمية التأكد بعدم إمكانية تخطي المستخدم لقواعد الأعمال إلزامية عند تحميل مؤسسة، ولكن يعطيك هذا أيضا الطريقة للسماح للمستخدم للوصول إلى معلومات محددة التي يتملكها مالك مخطط قاعدة البيانات فقط.

تبسيط الوصول للبيانات: من الممكن أن تقوم الإجراءات المخزنة بمعالجة البيانات قبل إرجاعها إلى تطبيق المستخدم. من الممكن أن يخفي هذا الأسلوب أي تعقيدات في نسق قاعدة البيانات وحتى يجعلها أكثر تأمين. ولسوء الحظ، من وجهة نظر المبرمج، هذه الخاصية ليست متطورة مثل SQLServer وSybase لمايكروسوفت. ويتم تطوير القدرة على إرجاع السجلات من الإجراءات المخزنة في نموذج مختلف وأكثر نفعا

لمستخدمي مدي منتج المطور ٢٠٠٠ لأوراكل، ولكن هذا الموقف يتغير ببطء كما سوف نشاهد في الفصول التالية من الكتاب.

استخدام ذاكرة متطور: في حالة تنفيذ نفس الإجراء المخزن بواسطة أكثر من مستخدم، تحتاج أوراكل إلى تخصيص ذاكرة لنسخة واحدة فقط من الإجراء. كل المستخدمون آنذاك يشتركوا في هذه النسخة وهذا يعني أن أوراكل من المكن أن يستفيد من أي ذاكرة متاحة لديك.

قابل للنقل: يمكنك نقل تعليمات البرمجة PL/SQL بين الحاسبات.ويعد هذا ذو أهمية قصوى لمستخدمي منتجات المطور لأوراكل لأنه يمكن نقل نفس تعليمات البرمجة بين طبقات التطبيق. (هذا يسمى تجزئة التطبيق). يمكنك أيضا نقل تعليمات البرمجة الإجرائية الخاصة بك إلى حاسبات خادم قاعدة بيانات أخرى لأوراكل.

الإجراء المخزن

أكثر برنامج فرعي شيوعا سوف تقوم بتكوينه هو الإجراء المخزن. يتم تحديد الإجراء المخزن PL/SQL بواسطة رأس إجراء بنفس طريقة تحديد إجراء الفيجوال بيسك باستخدام جملة SUB. وتسمى PL/SQL قطعة لغة هيكلية. ويتمك تحديد نقطة بواسطة قسم الإعلان وجملتي BEGIN وEND وقطعة Exception لمعالجة الخطأ الاختياري (يراجع التعليمات البرمجية ٢١٩. من الممكن أن تحتوي كل قطعة PL/SQL أخرى بداخلها.

تعليمات برمجية ٩-١

```
Declarations
. . . .
BEGIN
. . . .
PL/SQL statements
EXCEPTION
. . . .
Error handling PL/SQL statements
END;
```

يطابق هيكل القطعة هذا كل رموز PL/SQL سواء كان إجراء مخزن ذو وظيفة أو زناد أو قطعة رموز غير معروفة. ولإجراء النموذج لذلك موضح في التعليمات البرمجية ٩-٢.

ملاحظة تعليمات البرمجة غير المعروفة عبارة عن مصطلح سوف تتقابل معه دائماً والبرنامج الفرعي المخزن عبارة عن قطعة مسماة من الرمـــوز، مســماة بالطبع باسم الإجراء ولكن يمكنك ربط الرموز المرتبطة كل منها مع الأخر في قطعة بدون اسم مخزن. والقطعة الغير معروفة المسماة هــذه قــد يتــم تنفيذها من فيجوال بيسك. ولابد أن تكون محاطـــة بجمـــل BEGIN/END. وسوف نغطي هذا الموضوع في الفصل الرابع عشر، وحـــدات الوصـــول للبيانات، عند استدعاء الإجراءات المخزنة من الفيجوال بيسك. ويتم تحديــد الزنادات أيضاً باستخدام قطعة PL/SQL الغير معروفة.

تعليمات برمجية ٩-٢

```
PROCEDURE movie_insert (
  p_movie_name       IN VARCHAR2,
  p_length           IN NUMBER,
  p_rating           IN VARCHAR2,
  p_year_of_release  IN NUMBER
  ) IS
/* This procedure inserts a new movie in the MOVIE table*/
/* Declare local variable */
v_id   NUMBER(10);
BEGIN
  SELECT movie_sequence.NEXTVAL INTO  v_id FROM dual;
  INSERT INTO movies VALUES (
    v_id, p_movie_name, p_length, p_rating, p_year_of_release,
    USER, SYSDATE);
END movie_insert;;
```

أشياء قليلة عليك ملاحظتها من هذا المثال وهي:

- يتضمــن عنــوان الإجــراء الكلمــة الأساســية PROCEDURE واســم الإجــراء (movie_insert) وقائمة بالمعاملات المكتوبة بين أقواس هلالية والكلمة الأساســية IS التي تقدم جسم الإجراء.
- والاصطلاح التالي هو SQL وجمل PL/SQL بالأحرف الكبيرة بينما المتبقي من الرموز بأحرف صغيرة.

- يمكن تضمين الملاحظات في تعليمات البرمجة المحاطة بمحددات */ و/*.
- يجب تحديد أي متغيرات متسعة للإجراء بعد ذلك.
- من الممكن أن تكون كل جمل PL/SQL متعددة السطور ويجب إنهائها بشولة منقوطة تماماً مثل لغة C.
- يتم تحديد نص تعليمات البرمجة أو أكثر، الذي يكون مرتبط بواسطة BEGINوجمل END.
- يجب إنهاء كل الإجراءات وتحديدات الوظائف بجملة END.

الكتل المتداخلة

[illegible]

```
Declarations

BEGIN
   local declarations

BEGIN
     PL/SQL statements

   EXCEPTION

      Error handling PL/SQL
statements local to the block
END;
PL/SQL statements

EXCEPTION
```

```
Error handling PL/SQL statements
END;
```

تكمن الميزة العظمى لاستخدام القطع المتداخلة في أن معالجة الاستثناء ذاتية. فأينما يوجد استثناء في قطعة، يتم تنفيذ معالج EXCEPTION ومراقبة البرنامج تترك القطعة. عندما تنتهي المراقبة من قطعة بعد خطأ، تنتقل المراقبة إلى قطعة خارجية.

هيكل الوظيفة

النوع الثاني من البرامج الفرعية هو الوظيفة. يتم تحديد الوظائف بنفس طريقة الإجراءات إلا أنهم يتضمنوا قيمة إعادة ونوع البيانات الخاص بها في العنوان كما هو موضح في التعليمات البرمجية ٩-٣

تعليمات برمجة ٩-٣

```
FUNCTION GET_MOVIE_ID
( p_movie_name IN VARCHAR2) RETURN NUMBER
IS
v_movie_id   number(6);
BEGIN
  SELECT movie_id INTO v_movie_id FROM movies
     WHERE movie_name = p_movie_name;
  RETURN v_movie_id;
EXCEPTION
  WHEN OTHERS THEN
     RETURN -1;
END;
```

هيكل الزناد

نوع البرنامج الفرعي النهائي هو الزناد وهو يختلف عن أي بناء آخر فلا يمكنه أخذ أي معاملات لأنه لا يتم استدعائه ولكن يتم تنفيذه بواسطة تغيير في صف الجدول. ولابد أن يحتوي الزناد على معلومات تعرف نوع الإجراء الذي سوف يتم تنفيذه (مثل الإدراج والتحديث في المثال) وسواء يتم تنفيذه قبل أو بعد الحدث وسواء كان لكل صف أو لكل حركة.

واختلاف رئيسي آخر عن الإجراءات هو أن الزناد يحتوي على نسخة قديمة متاحة من صف البيانات وكذلك النسخة الحديثة (متاحة مع السجلات المنضمة) التي تعني أنه يمكنك فحص قيم العمود قبل تحديث الصفوف بالقيم الجديدة. يمكنك على سبيل المثال، التأكد من مشاهدة ما إذا كان القيمة الرقمية تتغير بواسطة أكثر من نسبة مئوية محددة. تعطي التعليمات البرمجية ٩-٤ مثال للزناد.

تعليمات برمجية ٩-٤

```
TRIGGER check_movie_length
BEFORE
INSERT OR UPDATE OF length ON movies
FOR EACH ROW
/*  The DECLARE statement can go here. */
BEGIN
  IF :new.length > 200 THEN
    RAISE_APPLICATION_ERROR(-20009,
'The film ' || :new.movie_name || cannot be over 200 minutes.');
  END IF;
END;
```

معاملات الإجراءات

يتم الدخول إلى والخراج من الإجراء المحزن عن طريق المعاملات فقط التي تقوم بتضمينها بين أقواس هلالية بعد اسم الإجراء. عندما تقوم بتحديد إجراء تقوم بتضمين معاملات شكلية للاستخدام داخل الإجراء. وعند استدعاءه، تقوم بتمرير معاملات فعلية إلى الإجراء. وسوف تلاحظ من الأمثلة أعلاه أن المعاملات الشكلية في الإجراء تحدد كل معامل باسمه ونوع البيانات وبينهما يمكنك تضمين كلمة أساسية اختيارية للطور تقوم بتحديد سواء كان معامل إدخال أو إخراج أو إذا كان يشارك هاتين الخاصيتين. وإذا قمت بحذفه، فسوف يكون الافتراضي IN. والقيم الممكنة كما في الجدول ٩-١.

الجدول ٩-١: أطوار المعاملات الممكنة

الطور	البيان
IN	هذا المعامل معامل مخرجات فقط ويمكن أن يمر إلى الإجراء المخزن ولا يمكن تغيير القيمة وهذا النوع هو الافتراض وهو الأكثر كفاءة بينما المعامل الفعلي قد يكون ثابت أو متغير.
OUT	هذا المعامل معامل مخرجات فقط ولا يمكنك تمرير قيمة إلى الإجراء المخزن بنوع المعامل هذا، وعليك فقط إعادة القيمة. ولابد أن يكون المعامل الفعلي متغير.
IN OUT	هذا المعامل قادر على تمرير قيمة إلى الإجراء المخزن وإعادة القيمة البديلة إلى إجراء الاستدعاء. ولابد أن يكون المعالم متغير.

وسوف تحقق إنجاز قليل في الأداء إذا أمكنك تجنب استخدام معاملات أكثر من اللازم. وعليك أن تكون حريص خاصة أنك لا تقوم بتمرير اسم أو وقت أو أي قيمة أخرى للمستخدم يمكن العثور عليها بسهولة من داخل الإجراء.

الحزم

كما أنه من المكن ترجمة إجراءات ووظائف الفيجوال بيسك في وحدة طبقية، من الممكن أيضاً ترجمة إجراءات ووظائف PL/SQL في حزم. وبنفس الطريقة سوف تستخدم لترجمة الرموز المرتبطة التي تتفق في الغرض سواء كانت بالنسبة للإجراءات التي تتطابق مع جدول معين أو كل الإجراءات التي تتطابق مع قسم معين، مثل الموارد البشرية أو الهندسة. وقد تريد أيضا وضع كل الرموز المرتبطة المساعدة في حزمة.

وإضافة غلى هذه الخاصية التنظيمية، يوجد سببان آخران لاستخدام الحزم. الأول، عندما يقوم أوراكل بسحب واحد من الإجراءات داخل ذاكرة SGA، فيقوم بسحب كل الحزمة.والفرص التي سوف تكون بحاجة إليها أكثر من إجراء واحد من الحزمة التي توجد في الذاكرة بالفعل، وهذا بوضوح يحسن الأداء.

السبب الثاني لاستخدام الحزم هو أنها الطريقة الوحيدة التي يمكن من خلالها استخدام نوع البيانات المحددة بواسطة المستخدم كمعامل للإجراء. وأكثر الأساليب المتقدمة التي سوف تتعرف عليها من خلال الفصل العاشر، تكوين الإجراءات المخزنة، تتطلب حزم.

تكوين الحزم

تشبه الحزمة الوحدة C: فهي تحتوي على مواصفات "أو عنوان| وجسم "تعليمات البرمجة الفعلية". ومعاً يقوما بتكوين الحزمة. ولتكوين حزم، نقوم بتكوين المواصفات والجسم كل على حدة. وفي باني الإجراء، عندما تقوم بتكوين موضوع جديد، عليك تحديد "Package (Spec)" للعنوان و"Package (Body)" للنص.

يحتوي الجسم على كل رموز PL/SQL الخاصة بالإجراءات و/أو الوظائف التي تجمع الحزم. تحتوي المواصفات على جمل Procedure من الجسم وأي تعريفات نوع محددة بواسطة المستخدم مستخدمة في المعاملات الفعلية وأي معاملات أخرى تريد أن تجعلها متاحة للحزمة ككل. وفي الفصل التالي سوف نقوم ببيان حزم متعددة التي تسهل من المتابعة في هذا الموضوع.

تلميح عندما تقوم بتحديد حزمة، تأكد أن تعريفات الإجراءات متطابقة تماماً في كلا الجسم والمواصفات. وإذا كنت غير متأكد، فمن الممكن أن تسبب هذه المراقبة لبس في رسائل الخطأ.

الوظائف المدمجة

بالإضافة إلى تكوين الحزم الخاصة بك تأتي حزم مبنية مسبقاً مع أوراكل. تزود هذه الحزم مبنية مسبقاً بإمكانات قوية لمستخدمي PL/SQL لكل شئ من إقفال جداول قاعدة البيانات إلى الوصول إلى حزم العمليات الخارجية من خلال ما يسمى بالأنابيب، كما أنهم يتضمنوا حزم لتكوين HTML. يحتوي الجدول ٩-٢ على بيان موجز عن بعض الحزم المدمجة المتاحة. ويجب أن تكون حذر على الأقل من هذه الحزم لأنه من الممكن أن يقوموا بإضافة بُعد آخر للتطبيقات الخاصة بك.

الجدول ٩-٢: حزم تم إنشائها من قبل

الحزمة	البيان
DBMS_ALERT	ويقوم بتمكين التأشير غير المتزامن للأحداث (يشبه الأحداث في الفيجوال بيسك).
DBMS_DDL	يسمح بأوامر لغة تحديد البيانات مثل تغيير أو ترجمة الإجراءات.
DBMS_DESCRIBE	إجراءات جداول PL/SQLفي الفواصل المحددة مسبقاً.وتعد هذه الحزمة مفيدة في ترتيب البرامج فرعية.
DBMS_LOB	يقوم بتزويد الحزم لمعالجة الوحدات الكبيرة (أوراكل ٨ فقط).
DBMS_LOCK	تمنح الوصول إلى إدارة الإقفال لأوراكل لتطوير مخططات قواعد بيانات الإقفال المعقد.
DBMS_OUTPUT	تقدم المخرجات الطرفية عن طريق الذاكرة الوسيطة
DBMS_PIPE	تربط بين العمليات المختلفة على نفس الكمبيوتر.
DBMS_SESSION	تتيح التحكم في خواص دورة التطبيق.
DBMS_SQL	تتيح تنفيذ SQL ديناميكية. وهذه هي لغة تحديد البيانات (DDL) وأوامر معالجة البيانات (DML) التي لم يتم تحليلها حتى وقت التشغيل.
DBMS_TRANSACTION	يزود بمراقبة حركة تشبه معادلي SQL.
DBMS_UTILITY	تزود بمنوعات من الإجراءات المساعدة مثل إجراءات الترجمة وجداول التحليل.
UTIL_FILE	تقوم بفتح وإغلاق وقراءة وكتابة ملفات نظام التشغيل.
UTIL_HTTP	تقوم بإرسال طلب محدد موقع الموارد العام إلى موقع ويب ويستلم الإجابة

ملاحظة

نظريا يجب أن تكون المدخلات إلى والمخرجات من قواعد بيانات SQL بواسطة SQL. تقوم PL/SQL ذاتها بكسر هذه القاعدة، ولكن كما تلاحظ من الحزم المتاحة، يوجد وظيفيا توجيه بديل لقاعدة البيانات. وإمدادات مثل هذه موجودة تقريبا في كل RDBMS وتضيف إلى الفعالية.

أدوات لتكوين الإجراء المخزن

توجد مجموعة من الأدوات التي تقوم بإدخال إجراءات ووظائف PL/SQL في قاعدة البيانات لأوراكل. وفي هذا الفصل، سوف نتعامل مع أكثر ثلاث في الأهمية:

- SQL*Plus
- Visual Database Tools
- Oracle Procedure Builder

أيهما تقوم باختياره فهو قرار شخصي، ولكن يجب أ، تعتاد على إمكانات الثلاث لكي يمكنك الحصول على قرار مبني على الإمكانات مقابل التكلفة.

SQL*Plus

SQL*Plus أساسا بيئة لكتابة جمل SQL وعرض النتائج. وبرنامج SQL*Plus موجود على كل حاسب موجود عليه أوراكل (الذي يغطي معظم الحاسبات الحديثة). على سبيل المثال، في بيئة ويندوز ٩٥، يمكنك تكوين دورة بواسطة تشغيل

C:\ORAWIN95\BIN\PLUS33W.EXE

أو أي شئ آخر مسمى قابل للتنفيذ بناءا على نوع الإصدار من البرنامج الموجود لديك. لاحظ أن الاتصال بقاعدة البيانات يكون عبر SQL*Net ولا يقوم بعمل أي شئ مع أي موارد DBC قمت بإعدادها.

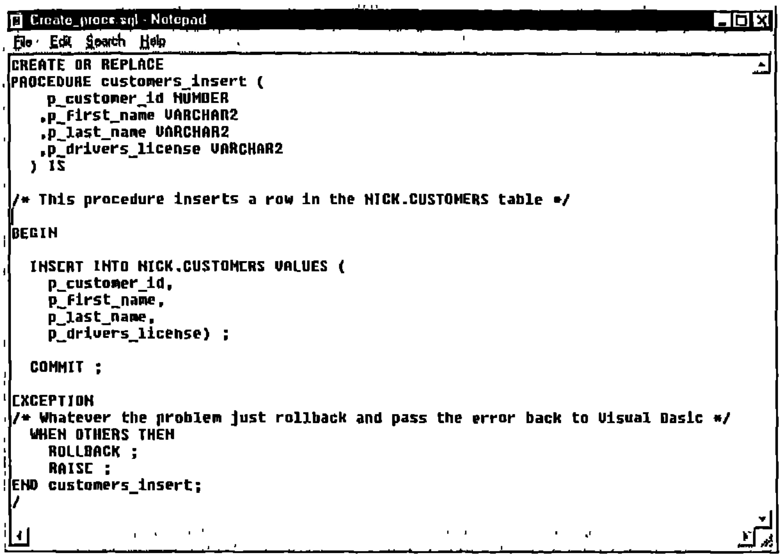
SQL*Plus، على كل حال أكثر من مجرد طريقة لإدخال جمل SQL. بالإضافة إلى احتوائها على إمكانات التزويد بتقرير بسيط لتوليد برنامج (قد تريد استغلال إمكانات التنسيق في دليل مستخدمي SQL*Plus لأوراكل)، من الممكن استخدامها لتكوين وتبديل وتنفيذ الإجراءات المخزنة.

يمكنك بسهولة تكوين ملف سكريبت SQL لتكوين الإجراء المخزن في أي محرر على أي خادم. يمكنك استدعاء محرر من موجه <SQL. والأمر الذي تقوم بكتابيه يعتمد على الحاسب

ولكن لابد أن يكون مسبوق بعلامة دولار. على سبيل المثال، إذا كنت تقوم باستخدام ويندوز ٩٥ أو NT، يمكنك كتابة SQL> $notepad.

التي سوف تقوم بالطبع بتشغيل محرر نوتباد. وإذا اتبعتها باسم ملف سوف تكتب هذا الملف. والاختلافات الوحيدة في السكريبت الفعلي عن ما تعرفنا عليه في القسم السابق هي أنه يجب الآن بالكلمات الأساسية Create Or Replace وتنهي السكريبت بشرطة مائلة أمامية كما في المثال بالشكل ٩-١.

الشكل ٩-١
تكوين سكريبت الإجراء المخزن في النوتباد.



```
CREATE OR REPLACE
PROCEDURE customers_insert (
    p_customer_id NUMBER
   ,p_first_name VARCHAR2
   ,p_last_name VARCHAR2
   ,p_drivers_license VARCHAR2
  ) IS

/* This procedure inserts a row in the NICK.CUSTOMERS table */

BEGIN

  INSERT INTO NICK.CUSTOMERS VALUES (
    p_customer_id,
    p_first_name,
    p_last_name,
    p_drivers_license) ;

  COMMIT ;

EXCEPTION
/* Whatever the problem just rollback and pass the error back to Visual Basic */
  WHEN OTHERS THEN
    ROLLBACK ;
    RAISE ;
END customers_insert;
/
```

تنفيذ السكريبت

يمكنك الآن تنفيذ ملف السكريبت من دورة SQL*Plus لتكوين الإجراء المخزن. على سبيل المثال، إذا كان ملف السكريبت create_proc.sql. فعليك تنفيذه بواسطة وضع علامة "@" قبل الاسم.

SQL> @c:\scripts\create_proc.sql

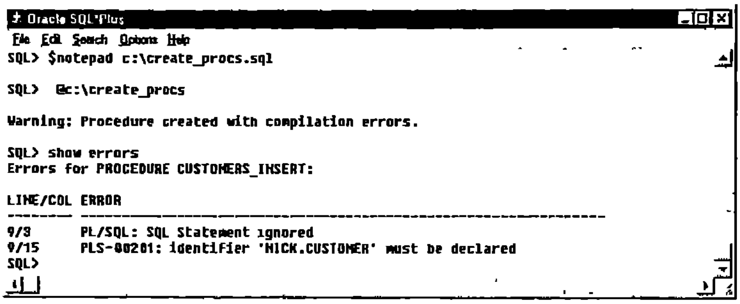
ملاحظة تضمين "OR REPLACE" كجزء من الأمر ليس بالضرورة يكون في أول مرة عند تكوين الإجراء المخزن ولكن قد تقوم بتضمينه لعمل ترابط.

تعتبر عملية تكوين إجراء مخزن سهلة ولكن صداقة المستخدم تنتهي هنا. فإذا كان أي أخطاء في تعليمات البرمجة، نحصل على رسالة خطأ مفهومة وتشير فقط إلى رقم السطر. وعندما تقوم

بتشغيل السكريبت لتكوين الإجراء ، قد تحصل على رسالة توضح وجود بعض الأخطاء عندما قامت PL/SQL بترجمتها. فإذا كان الوضع على هذه الحالة يمكنك عرض الأخطاء بواسطة كتابة

SHOW ERRORS;

وفي المثال أعلاه، My_Table لم يتم خروج الجدول وكذلك النتيجة موضحة في الشكل ٩-٢.



الشكل ٩-٢ توضح دورة SQL*PLUS أحد الأخطاء التي حدثت بعد حفظ أحد الإجراءات المخزنة.

تنفيذ الإجراء المخزن

لتنفيذ الإجراء المخزن من دورة SQL*Plus، استخدم

EXEC procedure_name (arguments);

إذا كنت في مدير الخادم سوف يترتب عليك استخدام الكلمة الأساسية بالكامل، EXECUTE.

وإذا كان الإجراء يتطلب معاملات OUT أو IN/OUT، فعليك إذا إعلان هذه المعاملات قبل استدعاء البرنامج الفرعي. وعليك إجراء ذلك بواسطة أمر VARIABLE SQL*Plus لإعلان متغير ربط كالآتي:

VARIABLE user_id NUMBER

أو

VARIABLE user_name VARCHAR2(25)

وعندما تقوم باستدعاء الإجراء المخزن، عليك أن تضع قبل متغير ربط فاصلة. والمشاهدة القيمة التي تم رجوعها، استخدم أمر PRINT. ومخرجات دورة SQL*Plus الموضحة في الشكل ٩-٣ تجعل هذا أسهل للفهم.

```
Oracle SQL*Plus
File Edit Search Options Help
SQL> VARIABLE user_name VARCHAR2(20);
SQL> EXEC test_out_proc (:user_name);

PL/SQL procedure successfully completed.

SQL> PRINT user_name

USER_NAME
--------------------------------
NICK

SQL>
```

الشكل ٩-٣

تنفيذ الإجـــراء المخـزن مــن SQL*Plus

وتقوم بعمل نفس الشيء لاختبار الوظيفة، كما هو موضح في الشكل ٩-٤.

```
Oracle SQL*Plus
File Edit Search Options Help
SQL> VARIABLE user_id NUMBER
SQL>  EXEC :user_id := test_out_fn('NICK');

PL/SQL procedure successfully completed.

SQL> PRINT user_id;

   USER_ID
----------
       103

SQL>
```

الشكل ٩-٤

تنفيذ الوظيفة في SQL*Plus.

معالجة أخطاء الإجراء

عند العمل مع PL/SQL، معالجة أخطاء تعليمات البرمجة المحددة للمعادل لـ Debug. جمل الطباعة في الفيجوال بيسك. وفي PL/SQL، يمكنك الاستفادة من الإجراء الخاص الموضوع في حزمة DBMS_OUTPUT، وداخل الإجراء تستخدم إجراء PUT_LINE

DBMS_OUTPUT.PUT_LINE (message);

على سبيل المثال، لإخراج القيمة من عدد المتغير، يمكنك كتابة

DBMS_OUTPUT.PUT_LINE ('The value of variable count is ' || count);

ولسوء الحظ، يتم عرض المخرجات على الشاشة عند إنهاء الإجراء. وعلى سبيل الافتراض، من الممكن أن تحتوي الذاكرة الوسيطة للمخرجات على ٢,٠٠٠ بايت التي يمكن استخدامها على وجه السرعة إذا كنت تقوم بمعالجة أخطاء كثيرة. يمكنك رفع هذه الكمية بواسطة استخدام أمر

التالي SQL*Plus

```
SET SERVEROUTPUT ON SIZE 10000;
```

وفي الحقيقة يمكنك إعداد الحجم على ١,٠٠٠,٠٠٠ حرف.

تلميح أيما أداة تقوم باستخدامها، تأكد أنك دائما تقوم بإدخال تعليمات البرمجة إلى ملف السكريبت. خاصة إذا كان أكثر من سطر واحد في الطول. وبهذه الطريقة، يكون لديك المصدر متوفر للتحرير وإعادة الاستخدام كلما تطور البرنامج.

أدوات قواعد البيانات المرئية

تعتبر أدوات قاعدة البيانات ضيف جديد على مجموعة الأدوات لتطوير الإجراء المخزنة لأوراكل. وتعتبر الأدوات ثورية في كونها أنها تقوم بعرض دعم مايكروسوفت لأوراكل بالرغم من التنافس المستمر. تحتوي أدوات قاعدة البيانات المرئية على بعض الميزات المفيدة وتأتي كجزء من حزمة Interdev ومع إصدار Enterprise من فيجوال بيسك. ففي فيجوال بيسك ٦، تم تكامل الأدوات إلى بيئة التطور، على الرغم أنهم قد يكون في حالة غامضة عند الاستخدام. ولقد تعرفنا أولا على مجموعة الأدوات هذه في الفصل ٦، تنفيذ قاعدة البيانات. هذه الأدوات كانت متاحة مع فيجوال بيسك ٥، ولكن تشغيلهم مع أوراكل تم تطويره على وجه السرعة.

بدء التشغيل

لكي تكون قادر أن تستخدم أدوات قاعدة البيانات المرئية من فيجوال بيسك، توجد طريقتان لعمل ذلك. يمكنك البدء من خلال بيئة البيانات إذا كانت توجد لديك واحدة بالفعل وطالما أنها مرتبطة بالمشروع. والطريقة التالية تستخدم ربط البيانات التي تعتبر مستقلة عن المشروع مما يجعلها طريقة عظيمة للتطوير.

١- افتح مشروع VB6 موجود أو جديد. (يمكنك تكوين ربط حتى لو لم تفتح المشروع).

٢- حدد نافذة Data View من قائمة View. وسوف يتم عرض نافذة عرض البيانات على جانب اليد اليمنى.

٣- انقر بزر الفأرة الأيمن على Data Links واختر Add a Data Link.

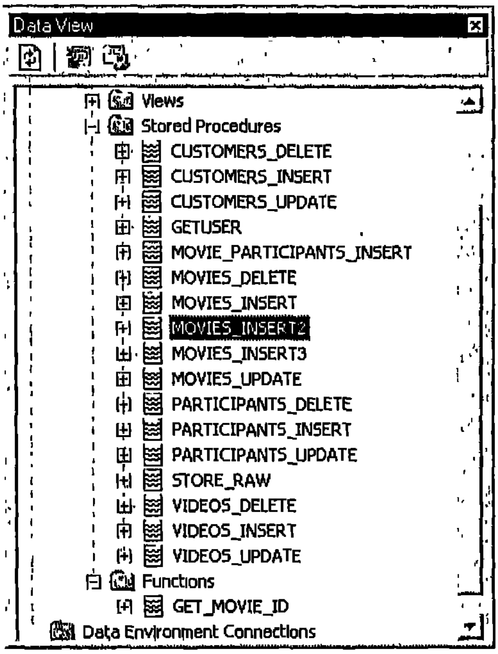
٤- أكمل خصائص Data Link (وعندما تتوصل إلى هذه الخصائص، سوف تدرك على وجه السرعة أن هذه الأداة مصممة لوحدات ActiveX Data). وعليك أولا تحديد مزود

OLE DB. وعلى علامة الجدولة Connection، اختر اسم مصدر البيانات من المربع المنبثق ثم املأ باسمك وكلمة المرور. ولاحظ أنه يجب استخدام مزود OLE DB لمايكروسوفت، على الرغم أن مزود OLE DB لأوراكل يبدو أنه يعمل جيدا.

٥- انقر على زر Test Connection لتتأكد أن كل شئ يمضي جيدا.

٦- انقر OK لتكوين الاتصال.

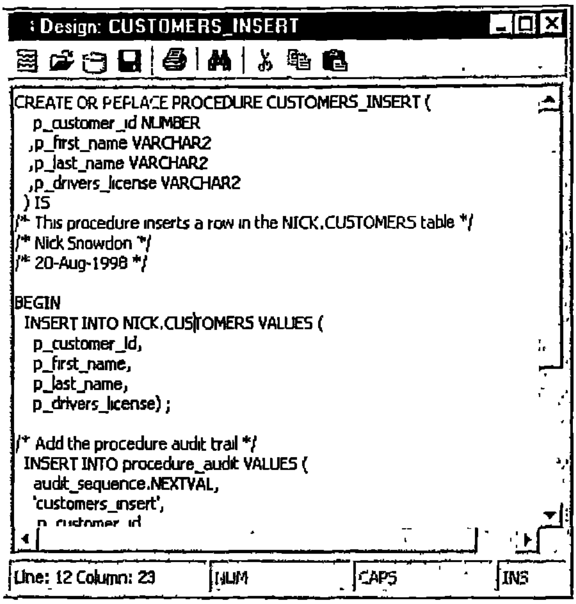
تضيف هذه العملية Data Link إلى البيئة الخاصة بك. اكتشف نافذة Data View الجديدة، وشاهد كم عدد الإجراءات والوظائف المخزنة التي يمكنك الوصول إليها في عرض قاعدة البيانات الخاصة بك (راجع الشكل ٩-٥).



الشكل ٩-٥
تصميم Visual Database Tools.

إدخال سكريبت

لإدخال إجراء مخزن جديد، انقر بزر الفأرة الأيمن على منطقة Stored Procedures من نافذة Data View ، واختر New Stored Procedure. وسوف تشاهد نافذة جديدة التي تعمل مثل نوتباد. إلا أنها تلون النص برموز وفقا للمعنى المقصود منها: فالتعليقات تكون بالون الأخضر والمسلسلات تكون بالون الأزرق والنص العادي يكون بالون الأسود. والكلمات الأساسية تظهر بألوان مختلفة. يمكنك الآن إدخال تعليمات البرمجة الإجرائية الخاصة بك (راجع الشكل ٦-٩). يعطيك محرر الإجراء تنسيق الإجراء للبدء به مما يجعل الاستخدام أسهل.



الشكل ٩-٦
إدخال سكريبت في VISUAL DATABASE TOOLS.

وعند الانتهاء من إدخال البرنامج الفرعي، يمكنك حفظه (انقر زر Save To Database)، وسوف يتم ترجمة وحفظ الإجراء في قاعدة البيانات. وفي حالة فشل الترجمة، سوف يتم إعادة رسالة الخطأ في مربع خطأ بوضوح، يكون غير مخزن على الخادم. وعندما يكون لديك ترجمة جيدة، يمكنك آنذاك الانتقال إلى مرحلة الاختبار فلا توجد دلالة في النافذة إذا كان قد تم ترجمة الإجراءات الموجودة بنجاح. يمكنك تحرير الإجراءات بواسطة النقر مرتان على الاسم في قائمة شجرة المستكشف.

تنفيذ الإجراء

لتنفيذ الإجراء، عليك التبديل إلى Visual Studio:

١-بدء تشغيل Visual Studio 6.

٢-تحديد وحدة بيانات جديدة.

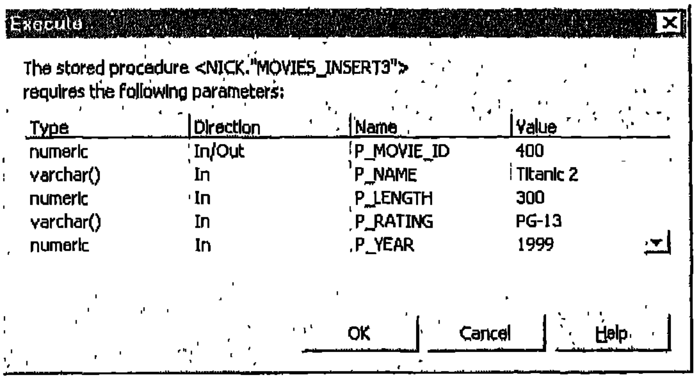
٣-تحديد مصدر بيانات ODBC.

٤-الدخول إلى أوراكل.

٥-فتح الاتصال في نافذة Data View لعرض كل الوظائف و الإجراءات المخزنة المتاحة.

٦-انقر بزر الفأرة الأيمن على الإجراء المخزن وحدد Execute.

ولاختبار الإجراء المخزن، يوجد شبكة، التي توضح المعاملات والأسماء والأنواع المطلوبة للإجراء مع مساحة لإدخال القيم الجديدة (راجع الشكل ٩-٦). لاحظ أنك لا تضع الحروف بين أقواس فردية. عند تنفيذ الإجراء، سوف تشاهد نافذة نص بالإجراء الذي تم تنفيذ بالفعل وقائمة بأي معاملات خارجة مع القيم الخاصة بهم. والشيء الغريب الوحيد هو أن البيئة لا يمكنها الإخبار عن عدد الصفوف المصابة . (وهذه ما سوف نشاهده من خلال طرق ODBC/OLE-DB). ومع هذا فإن سهولة الاستخدام تجعلنا نستهوي Visual Studio عن Visual Basic Data View. وهذه العملية تعمل جيدا أيضا لوظائف الاختبار.



الشكل ٩-٧ تنفيذ الإجراء المخزن في Visual Database Tools

ملاحظة يعتبر Visual Studio Data View منتج من Visual Basic Data View. يقوم المحررين باستخدام مخططات ترميز بالألوان المختلفة وكما شاهدنا في المثال، خواص كثيرة مفقودة من إصدار VB. فإذا كنت تقوم بفتح Visual Studio بأي طريقة، فإذا هذه هي البيئة التي تعمل منها.

إجمال أدوات قاعدة البيانات المرئية (Visual Database Tools)

إنه من البحث عن دعم جيل PL/SQL لمايكروسوفت بهذه الطريقة، وبالنسبة لـ Visual Database Tools لها بعض الميزات المفيدة. ومن الأسهل جدا استخدمها من SQL*Plus، ولكنها تفتقر إلى الخصائص المهمة.

فمن الناحية الإيجابية، تعتبر Visual Database Tools سهلة لاستخدام الواجهة التي يمكن استخدامها لقواعد بيانات مختلفة. فمن السهل تشغيل إجراء مخزن: فلا يجب تذكر أي صيغة لتنفيذها وتمرير المعاملات سهل وليس عليك تحديد معامل إخراج وتظهر قيمته آليا في نافذة المخرجات، وهذه ميزة مفيدة. وتظهر البيئة بشكل متعاون لمستخدمي أوراكل من الإصدارات السابقة.

وعلى كل، فإن Visual Database Tools لا تفتقر إلى بعض السمات الهامة التي سوف نشاهدها في Oracle Procedure Builder، والأكثر ملاحظة القدرة على معالجة الأخطاء للإجراء. وعلى وجه الشك في إمكانية بناء هذه الميزة في منتج مايكروسوفت. أيضا Visual Database Tools تفتقد أيضا القدرة على معالجة الحزم التي تكون قاعدة لتطبيقات PL/SQL الهامة.

باني الإجراء لأوراكل (Oracle Procedure Builder)

يأتي Oracle Procedure Builderكجزء من مجموعة Developer 2000 لأدوات التطبيق لأوراكل، ولكنه متاح أيضا كحزمة منفصلة. وإذا كانت الشركة التابع لها لا تستخدم Developer 2000، فإن هذا البديل يزود أكثر تكلفة وتعتبر جزءا الحقيقة أنه يعتبر البيئة المطورة كاملة الخصائص والوحيدة المتاحة لـ PL/SQL. فمن السهل نسبيا التحميل من CD مستخدما برنامج التحميل القياسي لأوراكل، orainst.exe.

وعند تشغيل Oracle Procedure Builder، يجب أولا استخدام خيار قائمة „ File Connect للاتصال بقاعدة البيانات باسمك وكلمة المرور الخاصة بك واسم قاعدة البيانات بالطريقة العادية. وإذا لم يكن قد تم إعداد المزود والخادم للاتصالات، إذا راجع الفصل الحادي عشر، تقنية جانب المزود. وعند تمام الاتصال بقاعدة البيانات، سوف تشاهد شاشة مثل الموضحة في الشكل ٩-٨.

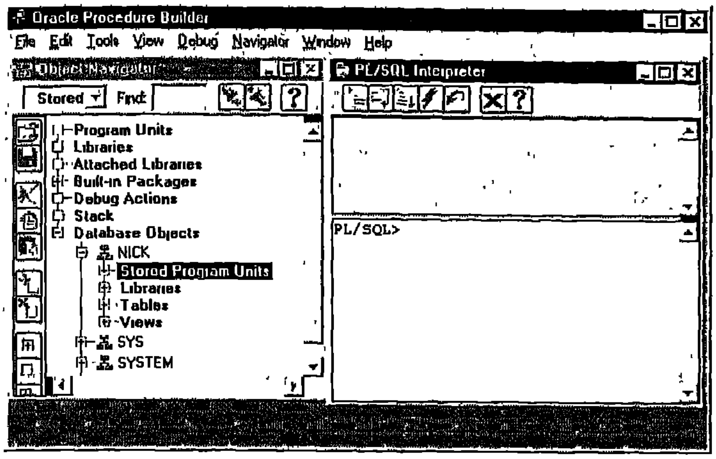
الخصائص التي يقدمها Procedure Builder كالآتي:

- عرض كل الوحدات المرئية للمستخدم، بما فيها البرامج الفرعية والجداول.
- تكون وتحرير الإجراءات والوظائف
- تكوين وتحرير الحزم
- تشغيل البرامج الفرعية المخزنة على الخادم

- مشغل (مفسر) PL/SQL المحلي الذي يتضمن الآتي:
  - القدرة على وضع نقاط توقف في تعليمات البرمجة
  - القدرة على مشاهدة قيم كل المتغيرات عند نقطة التوقف.
  - القدرة على الانتقال داخل تعليمات البرمجة سطر بسطر

الشكل ٩-٨

منطقة عمل Oracle Procedure Builder.



ملاحظة

قد يتغير اسم Procedure Builder إلى Application Builder لـ PL/SQL. يعكس هذا الاتجاه إلى تضمين لغات أخرى مثل جافا في قاعدة البيانات لأوراكل.

استخدام Procedure Builder

يعتبر Procedure Builder بيئة قوية لتطوير واختبار وصيانة الإجراءات المخزنة والوظائف. سوف يعطيك هذا القسم مقدمة إلى استخدامه وسوف تغطي بما فيه كفاية إنتاج إجراءات على مستوى الإنتاج. ومع هذا، لأن البيئة متسعة جدا، لابد أن تكون حذر أنه يوجد إمكانات متعددة أكثر سوف يتم تغطيتها.

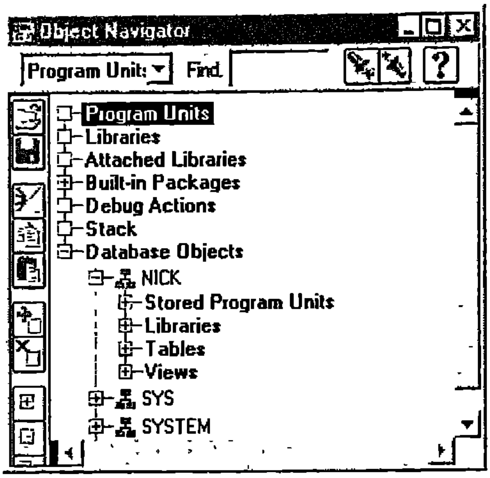
توجد ثلاث نوافذ رئيسية سوف تقوم باستخدامها في Procedure Builder. وإذا كانت هذه غير مرئية عند استخدام البرنامج، فإنه يمكنك جعلهم مرئيين من قائمة Tools. وسوف يتم تغطية كل من هذه النوافذ في الترتيب العادي الذي تريد ظهورهم فيه.

مستكشف الوحدة (Object Navigator)

نافذة Object Navigator هي نقطة البداية لأي من العمليات الخاصة بك والموضحة في الشكل ٩-٩. وتكون دائما حاضرة في التطبيق. ويجعلك مع مخطط مثل الشجرة بكل الوحدات التي يمكنك الوصول إليها. تسمى وحدات المستوى الأعلى نقاط توصيل ونقاط التوصيل التي يمكن لك الوصول إليها موضحة في الجدول ٩-٣. وفي الشكل ٩-٩، يمكنك مشاهدة نقطة توصيل وحدات قاعدة البيانات تم اتساعها لعرض البرامج الفرعية.

الشكل ٩-٩

نافذة Object Navigator.



الجدول ٩-٣: مخطط نافذة Procedure Builder Navigator

نقطة التوصيل	البيـان
وحدات البرنامج	تعرض الإجراءات التي يتم تخزينها محليا. تستخدم هذه الإجراءات لتشغيلها في معالج الأخطاء ويمكن تحويلها فيما بعد إلى الإجراءات المخزنة في قاعدة البيانات.
Libraries	تتيح لك تكوين وتعديل مكتبات الإجراءات. وسوف يتم التعامل مع هذه الأشياء في هذا الكتاب.
Attached Libraries	تتيح لك إرفاق وفصل مكتبات PL/SQL الموجودة.
Built-in Packages	تتيح لك مراجعة الحزم المدمجة المتاحة
Debug Actions	تتيح لك مراقبة إجراءات معالج الأخطاء.
Stack Used in Debug mode	ويتيح لك عرض قيم كل المتغيرات الحالية في مجال وحدة تنفيذ البرنامج.
Database Objects	ويتيح لك عرض وحدات قواعد البيانات المختلفة التي يمكنك الوصول إليها مثل الجداول، والإجراءات والزنادات. إذا قمت

الجدول ٩-٣: مخطط نافذة Procedure Builder Navigator

نقطة التوصيل	البيان
	بعرض قائمة وحدات البرنامج المخزن، توجد علامة نجمية بطول الاسم تدل على أن تعليمات البرمجة في حالة غير الترجمة ويحتاج إلى انتباهك.

تكوين وحدة البرنامج المحلي

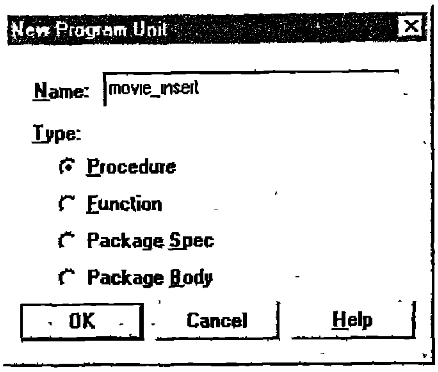
النوع الأول من إجراءات PL/SQL الذي سوف نقوم بتكوينه هو وحدة البرنامج الذي يعد يتم تنفيذه على محطة العمل وليس على خادم قاعدة البيانات. وعندما تقوم بإخراج باني الإجراء، فأنت تفقد تعليمات البرمجة، لذلك إذا كنت تريد الاحتفاظ به، عليك نقله إلى الإجراء المخزن، العملية التي سوف تشاهدها فيما بعد في هذا القسم بهذا الفصل "محرر وحدة البرنامج المخزن". والطريقة البديلة للاحتفاظ بوحدات البرنامج هو استخدام Edit/Export. وسبب الحاجة إلى وحدات برامج محلية مطلقا هو استخدام معالج الأخطاء.

تحذير وحدات البرنامج في إصدار Procedure Builder 1.x لا يمكن تدعيم الفعالية المتطورة مثل جداول وخلايا PL/SQL. وهذه الفعالية متاحة في الإصدار ٢.

لتكوين وحدة البرنامج، استخدم الخطوات التالية.

١-انقر نقطة توصيل وحدات قاعدة البيانات.

٢-انقر زر Create. سوف يقوم Procedure Builder بعرض مربع الحوار لتعريف نوع تعليمات البرمجة التي تريد تكوينها. تتضمن الخيارات الخاصة بك الإجراءات والوظائف ومواصفات الحزم والجسم (راجع الشكل ٩-١٠).

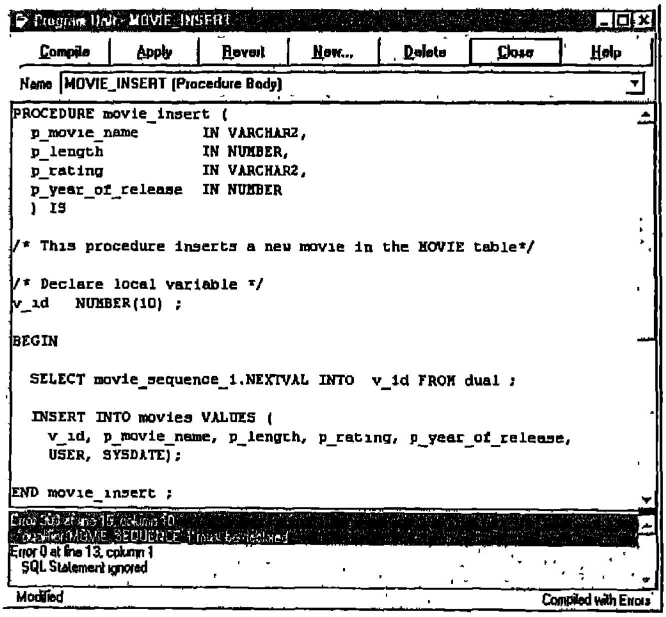


الشكل ٩-١٠ اختيار أي نوع من وحدات الرموز لتكوينها

٣-انقر الإجراءات واملأ الاسم لوحدة PL/SQL. في هذه الحالة، سوف نقوم بإدخال movie_insert. لاحظ أنه يوجد علامة نجمية بعد اسم وحدة البرنامج في مربع نص الاسم. وهذا يعني ترجمة الوحدة.

٤-ادخل الإجراء وانقر زر Compile أعلى النافذة.

٥-راجع أي رسائل خطأ وسوف تظهر هذه في النصف الأسفل من النافذة كما موضح في الشكل ٩-١١. وإذا قمت بالنقر على أي رسالة خطأ، سوف يتم وضع الخلية في السطر الزائد في النصف الأعلى من النافذة. في هذه الحالة، المشكلة أنه لم يتم تحديد movie_sequence_id. (تم استخدام الاسم الخطأ).



الشكل ٩-١١ تنفيذ رسائل الخطأ بعد الترجمة.

٦-عندما تقوم تعليمات البرمجة بالترجمة، سوف تختفي العلامة النجمية من مربع نص الاسم. ويمكنك الآن تشغيل وحدة البرنامج.

سوف تلاحظ عدم وجود شريط تمرير أفقي في المحرر. يبدو هذا غريب بشكل غير ملائم، والطريقة الوحيدة حول ذلك هي استخدام الفأرة لتمييز النص على حافة النافذة وتدعيم التمرير الأفقي. ولهذا فهو يشجعك على إنتاج سطور قصيرة من تعليمات البرمجة.

تحذير وحدات البرنامج المحلية تعتبر إجراءات محلية غير مخزنة على خادم أوراكل وبمجرد إنهاء Procedure Builder، سوف يتم فقد وحدة البرنامج إذا لم يتم حفظها بـ Edit/Export.

تشغيل تعليمات البرمجة باستخدام مفسر PL/SQL

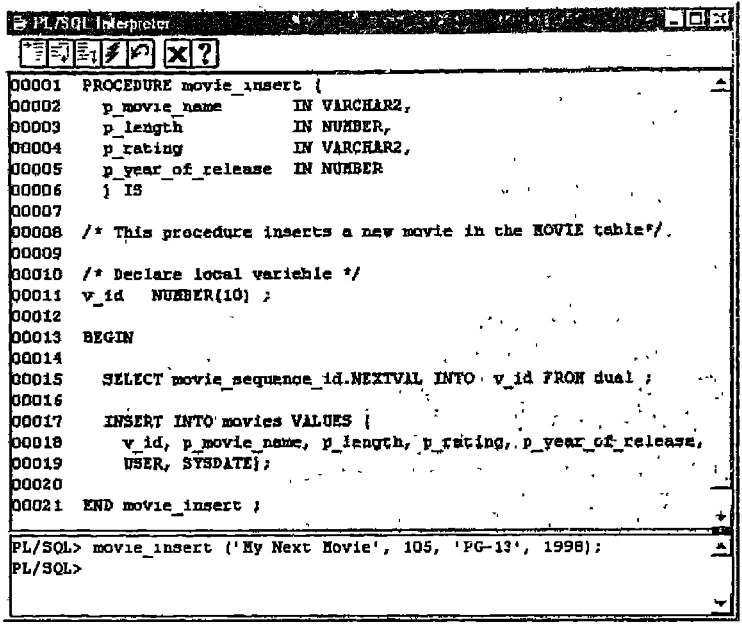
والآن قمنا بتكوين وحدة البرنامج، وأفضل طريقة لاختباره هي مفسر PL/SQL، الذي يتيح لك ليس فقط تشغيل تعليمات البرمجة ولكن أيضا تصحيح الأخطاء له. إذا كان مفسر PL/SQL غير مرئي، عليك اختياره من قائمة Tools.

يقدم لنا مفسر PL/SQL إمكانية انقسام الشاشة. إذا كان يوجد لديك وحدة برنامج في محرر وحدة البرنامج، فسوف تظهر تعليمات البرمجة آليا في النصف الأعلى من النافذة كما هو موضح في الشكل ٩-١١، وإذا لم يكن يوجد لديك، عليك إذا إغلاق محرر وحدة البرنامج ثم افتحه ثانية بواسطة النقر مرتين على البرنامج في Object Navigator.

ولتنفيذ الإجراء من Procedure Builder، عليك إدخال اسم الإجراء في النصف الأسفل من نافذة PL/SQL Interpreter وتضمين أي معاملات.

```
PL/SQL> movie_insert ('My Next Movie', 105, 'PG-13', 1998);
```

يمكنك مشاهدة ذلك في الشكل ٩-١٢.



الشكل ٩-١٢ تنفيذ وحدة البرنامج في PL/SQL Interpreter.

معالجة معاملات المخرجات

لقد شاهدنا أنه يمكن التعامل مع معاملات المخرجات بواسطة تضمينهم في استدعاء الإجراء، ومع هذا فإن معاملات المخرجات أكثر صعوبة في التعامل. فإذا كان يوجد لديك معاملات مخرجات، فأنت بحاجة لتكوين متغير محلي بواسطة أمر CREATE وبعد إكمال الإجراء، يمكنك عرض القيمة باستخدام TEXT_IO.PUT_LINE كالآتي:

```
PL/SQL> .CREATE NUMBER out_parm1
PL/SQL> .CREATE CHAR out_parm2
PL/SQL> test_proc (in_parm, :out_parm1, :out_parm2);
PL/SQL> TEXT_IO.PUT_LINE (TO_CHAR(:out_parm1));
PL/SQL> TEXT_IO.PUT_LINE (:out_parm2);
```

كما يتم التعامل مع المنهج نفسه من خلال الوظائف:

```
PL/SQL> .CREATE NUMBER return_val1
PL/SQL> :return_val1 := test_function ('My Movie');
PL/SQL> TEXT_IO.PUT_LINE (TO_CHAR(return_val1));
```

ففي كل حالة، لاحظ استخدام علامة الترقيم قبل المتغيرات. وعلامة الترقيم هذه تدل على أنك تتعامل مع متغير ربط، بمعنى، المتغير الذي يمكن لمعاملات المخرجات والقيم المردودة الربط

به. ويجب ملاحظة أنه لطباعة قيمة رقمية، فأنت بحاجة أولا لتحويلها إلى سلسلة حرفية بوظيفة TO_CHAR.

تحذير في إصدارات Procedure Builder قبل إصدار ٢، يجب استخدام معاملات IN OUT بدل من معاملات OUT لهذا لكي تعمل.

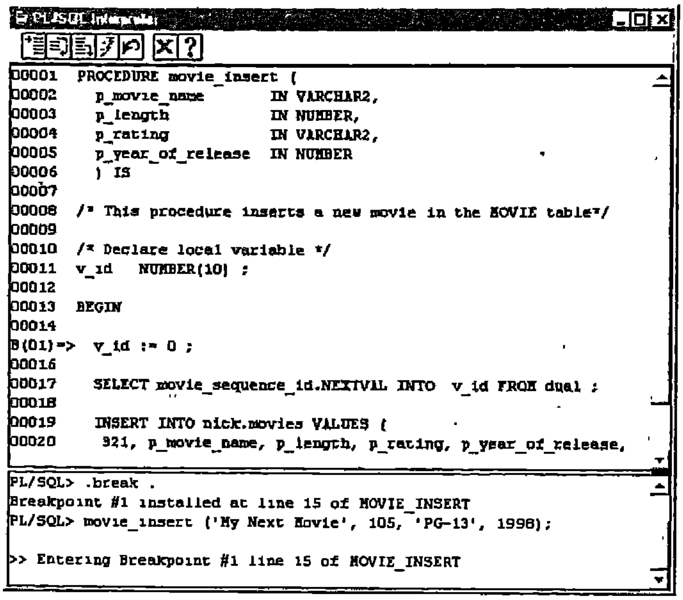
معالجة أخطاء الإجراء الخاص بك

ما لم تكن واحد من هؤلاء القلة الذين لا يعملون أخطاء، عليك أن تدرج في التدريب معتادا على تسهيلات معالجة الأخطاء الموثوق بها المتاحة الآن في فيجوال بيسك. فلا أتخيل الحال بدونهم، وإذا كنت أقوم باستخدام بيئة برمجة لا تزود بمثل هذه الميزة الفنية، فسوف يكون الوضع صعب. والحقيقة أن Procedure Builder لا يزود هذا المستوى من معالجة الأخطاء. ومن الناحية الأخرى، البدائل (SQL*Plus وVisual Database Tools، على سبيل المثال) لا تزود بأي معالجة أخطاء مطلقا.

ولتكوين دورة معالجة أخطاء لوحدة البرنامج، نواصل ما تركناه في هذا القسم "تشغيل تعليمات البرمجة باستخدام PL/SQL Interpreter ". توجد خاصيتان رئيسيتان لمعالج الأخطاء: نقاط التوقف وفحص القيم والمتغيرات.

ولتكوين نقطة توقف، انظر في النصف الأعلى من نافذة PL/SQL Interpreter: وهنا تستطيع مشاهدة تعليمات برمجة المصدر الذي تقوم بتكوينه. وإذا قمت بالنقر مرتين على أي سطر من الرموز، سوف يتم وضع نقطة توقف هناك، ويمكنك التأكد من ذلك لأن رقم السطر يتغير إلى "B" متبوعة برقم نقطة التوقف. سوف تلاحظ أيضا أن هذا منعكس كرسالة في النصف الأسفل من النافذة (راجع الشكل ٩-١٣).

الشكل ٩-١٣: إعداد نقاط التوقف للإجراء.

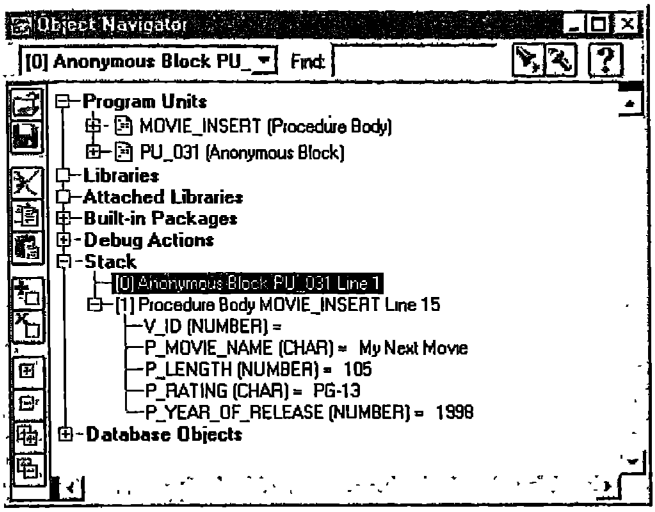


تحـذيـر عليك النقر مرتان على جملة PL/SQL القابلة للتنفيذ. هـذا يعنـي أنـه لا يمكنك وضع نقطة توقف في جملة SQL، مثل INSERT. وفي حالة إجراء ذلك، سوف تحصل على رسالة PDE-CBR001. وإذا كان في هذا مشكلة، عليك وضع سطر افتراضي مثل الموضح في السطر ١٥ بالشكل ٩-١٣.

وعند بدء البرنامج بالطريقة العادية، يمكنـك اسـتخدام شـريط الأدوات أعلـى PL/SQL Interpreter لكي:

- تدخل في استدعاء البرنامج الفرعي
- تتخطى استدعاء البرنامج الفرعي
- تخرج من البرنامج الفرعي لبرنامج الاستدعاء
- تنتقل إلى (بمعنى تنفيذ البرنامج حتى تتم مواجهة نقطة التوقف التالية)
- إعادة الإعداد لتبديل مستوى معالجة الأخطاء

عندما يصل برنامج التنفيذ نقطة التوقف، يمكنك عرض قيم كل المتغيرات في الطبقة. والطبقة تعتبر واحدة من نقاط التوصيل في Object Navigator، وتقم بعرض القيمة الحالية لكل متغـير في البرنامج (راجع الشكل ٩-١٤).



الشكل ٩-١٤

تتيح لك Stack مراقبة كل قيم المتغيرات في البرنامج.

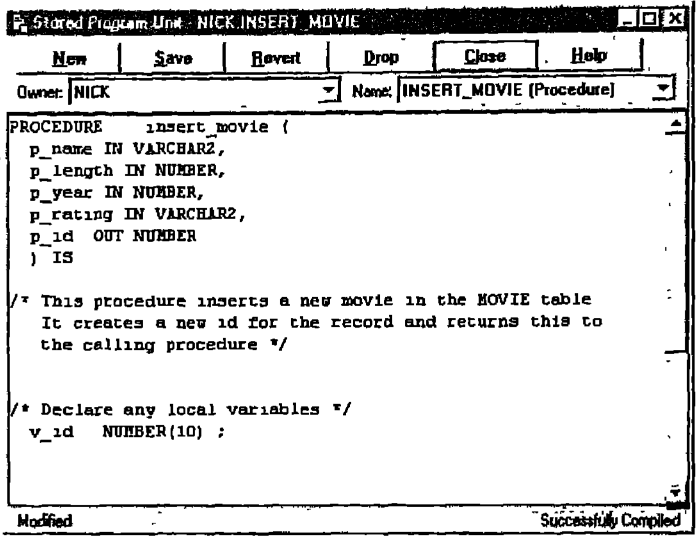
محرر وحدة البرنامج المخزن "Stored Program Unit Editor"

بمجرد الانتهاء من اختبار تعليمات البرمجة في معالج الأخطاء، يمكنك الآن تخزينها على خادم قاعدة البيانات كإجراء مخزن. ولعمل هذه الخطوة، عليك قص ثم لصق تعليمات البرمجة من Program Unit Editor إلى Stored Procedure Editor بنفس طريقة تكوين وحدة البرنامج، ماعدا في هذه المرة تنقر مبدئيا على مستوى Stored Procedures أسفل نقط توصيل Database Objects في Object Navigator.

يختلف نسق Stored Procedure Editor قليلا عن Program Unit Editor. وكما تشاهد في الشكل ٩-١٥، الاختلاف الرئيسي أن الآن تستخدم زر Save بدل من زر Compile.

ملاحظة

يمكنك استخدام Stored Program Unit Editor مثل البيئة العاملة الخاصة بك إذا كنت تريد ذلك ولكن تذكر أنه لا يمكنك استخدام معالج الأخطاء على الرموز.



الشكل ٩-١٥ Stored Program Unit يشبه Program Unit Editor.

مكان تخزين تعليمات البرمجة

من الأشياء الغريبة عن البرامج المخزنة أنه إذا قمت بتحرير إجراء يعمل ولا يقوم بالترجمة، بينما الإجراء الأسبق ما زال يعمل. هذه السمة مفيدة إذا كان عليك عمل التغييرات لقاعدة بيانات حية. يتيح لك أيضاً إدراك أن تعليمات برمجة المصدر يتم تخزينها بشكل منفصل عن تعليمات برمجة الترجمة.

وللبحث عن تعليمات برمجة المصدر، يمكنك زيارة العرض الخاص المسمى USER_SOURCE، وعليك عرضه بالجملة التالية:

```
SELECT name, text FROM user_source
ORDER BY name, line;
```

إذا كنت شيء ما مثلي، قد تكون هذه المعلومات ضارة لوقتك، يمكنك تخيل كتابة المحرر الخاص بك بالخصائص التي تريدها وبرنامج الفيجوال بيسك يستطيع اختيار كل الإجراءات الخاصة بك.

معالجة الأخطاء في PL/SQL

تحتوي الإجراءات المخزنة على نموذج واحد من معالجة الأخطاء، الذي يكون من خلال قطعة رموز التي تم بدئها بطريقة مشابهة معالج أخطاء "On Error Goto". في هذه الحالة، المساوي

لجملة GOTO يتم استدعاءه آليا إذا قمنا بتزويد قطعة المعالجة EXCEPTION. هذه قطعة رموز يتم تحديدها بواسطة الكلمة الأساسية EXCEPTION، ومثال ذلك موضح في التعليمات البرمجية ٩-٥. ونقول بأن PL/SQL تقوم بإظهار استثناء عند مواجهة خطأ.

تعليمات برمجة ٩-٥

```
/* This is the EXCEPTION block */
EXCEPTION
  WHEN NO_DATA_FOUND THEN
    code := 1;
  WHEN VALUE_ERROR THEN
    code := 2;
  WHEN OTHERS THEN
    code := 99;
```

في حالة حدوث أخطاء في البرنامج، يتم مرور المراقبة آليا في قطعة Exception، ويمكنك التقرير حول كيفية الاستمرار. ومن الأشياء المعتادة التي يمكنك عملها هو تجريب الأخطاء الممكنة المختلفة ثم تقوم بتزويد تعليمات برمجة لمعالجة برنامج التدريب المطلوب للإجراء لكل. وقبل القيام بهذا، يجب النظر أولا إلى أنواع الأخطاء المدمجة، وهذه الأخطاء موضحة في الجدول ٩-٤.

الجدول ٩-٤: أنواع الأخطاء المدمجة في PL/SQL

الاستثناء	البيان
CURSOR_ALREADY_OPEN	كانت هناك محاولة لفتح الخلية مفتوحة بالفعل. عليك إغلاقها أولا أو استخدم أخرى مختلفة.
DUP_VAL_ON_INDEX	كانت هناك محاولة لحفظ السجل بمفتاح فريد موجود بالفعل.
INVALID_CURSOR	عملية غير صحيحة في الخلية.

الجدول ٩-٤: أنواع الأخطاء المدمجة في PL/SQL

الاستثناء	البيان
INVALID_NUMBER	خطأ في سلسلة لتحويل الرقم لأن السلسلة لا تمثل رقم صحيح.
LOGON_DENIED	معرف أو كلمة مرور مستخدم غير صحيح عند الدخول (بنفس الطريقة غير مطابق لاستدعاءات برنامج الفيجوال بيسك).
NO_DATA_FOUND	نتيجة جملة SELECT INTO لم تقوم بإعادة صفوف.
NOT_LOGGED ON	معرف أو كلمة مرور مستخدم غير صحيح عند الدخول (بنفس الطريقة غير مطابق لاستدعاءات برنامج الفيجوال بيسك).
PROGRAM_ERROR	مشكلة داخلية لأوراكل
ROWTYPE_MISMATCH	تخصيص الخلية بينما تكون الخلايا من أنواع مختلفة.
STORAGE_ERROR	خطأ ذاكرة.
TIMEOUT_ON_RESOURCE	انتهاء الوقت أثناء انتظار مورد النظام.
TOO_MANY_ROWS	نتيجة جملة SELECT INTO تقوم بإعادة أكثر من صف واحد.
VALUE_ERROR	وجود خطأ تحويل أو حسابي ما. قد يكون هذا أيضا نتيجة حجم العمود الذي تم تجاوزه.
ZERO_DIVIDE	تقسم على خطأ صفر

ملاحظة بعد اختبار تعليمات برمجة الرجوع باستخدام WHEN...THEN، يجب أن يكون لديك سطر رموز واحد قابل للتنفيذ على الأقل. إذا كنت تريد تجاهل الخطأ، عليك وضع سطر تعليمات البرمجة هناك حتى لو لم يقم بتنفيذ الكثير.

شئ واحد لابد أن تكون حذر منه هو إذا لم تقم بتصيد الخطأ بقطعة EXCEPTION، فسوف يعود الخطأ للبرنامج الفرعي للاستدعاء (وهذا مرة أخرى يشبه طريقة عمل الفيجوال بيسك). هذا يعني معالجة الأخطاء في الفيجوال بيسك وهذا قد يتطلب تنفيذ بعض SQL أكثر أو ربما إجراء PL/SQL آخر لإلغاء تعديلات الحركة الجزئية وهذه ليست أكفأ طريقة لعمل الأشياء.

ومن الناحية الأخرى، إذا قمت بتصيد الخطأ في PL/SQL، فلن يكون إجراء الاستدعاء مدرك بحدوث أي أخطاء. ولحسن الحظ، من الممكن إعادة تأشير الخطأ بحيث يشعر برنامج الاستدعاء بحدوثه ويبلغ المستخدم (بنفس طريقة مربع الرسائل). وفي برنامج الاستدعاء، يمكنك التأكد أن حتى إذا تم حدوث خطأ، فإنه تم التعامل معه وليس عليك اتخاذ أي إجراء تصحيح ما عدا إبلاغ المستخدم. وإعادة تأشير رسالة الخطأ هذه يتم تنفيذها ببساطة بواسطة جملة RAISE.

```
EXCEPTION
  WHEN OTHERS THEN
    ROLLBACK;   /* Rollback changes if situation warrants it */
    RAISE;      /* Notify the calling routine */
```

يفضل اعتبار هذا التقريب كحد أدنى عند اكتشاف الأخطاء وعليك إضافة جملة ROLLBACK للتراجع عن أي بيانات تم إدخالها أو تحديثها أو حذفها ثم يمكنك بعد ذلك كتابة تعليمات البرمجة لفيجوال بيسك لإعلان المستخدم بعدم وجود بيانات تم حفظها نتيجة الخطأ.

تلميح

يجب أن يكون لديك عادة التأكد أن كل الإجراءات المخزنة يتم استدعائها من برامج فرعية خارجية مثل الفيجوال بيسك ويعالج حالات الخطأ داخليا. وبهذه الطريقة، لن يكون عليك التزويد بمعالجي أخطاء متقدمة في كل تطبيق يستفيد منهم. وتذكر أن أي إجراء تقوم بكتابته يمكن أن يتضمن خطأ.

تدعيم رسالة الخطأ

سيقوم الإجراء الذي تم تخزينه بإرجاع رسالة الخطأ لأوراكل إلى برنامج فيجوال بيسك الخاص بك، الذي سيحتاج بدوره إلى برنامج فرعي بمعالجتها (انظر فصل ١٣، موضوعات تصميم فيجوال بيسك لمزيد من المعلومات). غير أن هذا الأمر يقوم بتحديد اختيارك الخاطئ إلى مجموعة من رسالات الخطأ المحددة من قبل وأرقام أوراكل المحددة فقط. وربما تكون بحاجة إلى إرسال الرسائل الخاصة بك مرة أخرى التي قد تكون أكثر تحديدا عن الرسائل الافتراضية.

فعلى سبيل المثال، إذا كنت تقوم بمحاولة تحديث سجل باستخدام معرف معين واكتشفت أن هذا السجل غير موجود، عندئذ يمكن إعادة رسالة الخطأ الافتراضية الخاصة بأوراكل التي تخبر بأن هذا السجل لم يتم العثور عليه. حيث يكون هذا الأمر في لغة أوراكل عبارة عن رسالة أكثر وضوحا التي تتيح للمستخدم معرفة أنه لم يتم العثور على المعرف كما تزوده بقليل من التفاصيل الإضافية التي ربما قد تشتمل على المعرف نفسه. وسوف يفضل المستخدمون لتطبيقك بالتأكيد هذه الطريقة الخاصة بالبحث عن خطأ ما!

وللقيام بإرجاع رسالة الخطأ، استخدم العبارة التالية:

```
RAISE_APPLICATION_ERROR (error_number, error message, flag);
```

ويجب أن يكون رقم الخطأ ما بين ٢٠٠٠٠ و٢٠٩٩٩ كما يتسع طول الرسالة إلى ٢٠٤٨ حرفا. وتكون العلامة اختيارية وغالبا لا يتم استخدامها : ويكون الوضع الافتراضي لها FALSE والتي تعني أن رسالتك قد حلت محل رسائل الخطأ الأخرى التي تم إرجاعها إلى الإجراء الحالي.

وفيما يل مثال يوضح كيفية تحديد رسالة الخطأ

```
IF value > test_value THEN
RAISE_APPLICATION_ERROR
(-20001, 'The value was greater than ' || TO_CHAR (test_value));
END IF;
```

وفي حالة محاولة الاستفادة عن طريقة معالجة الأخطاء هذه ، يجب الاحتفاظ بمسار شروط الخطأ عن طريق حفظها في جدول صغير مع تعليمات البرمجة خطأ لاستخدامه في الإجراء بالإضافة إلى رسالة ورقم الخطأ المرتبط . عندئذ يمكن استخدام هذا الأمر خلال قاعدة البيانات بأكملها. وهناك طريقة بديلة تتمثل في تخزين رسائل وأرقام الصفحات كثوابت في عنوان الحزمة. أيا كانت الطريقة التي تختارها لمعالجة الأخطاء، يجب التأكد من أن استخدام أرقام الأخطاء يتم التحكم فيه بدرجة كبيرة.

تلميح من المفضل ليس فقط وصف الأخطاء بالقدر الذي تستطيعه ولكن أيضا إعطاء تفسير لما يعنيه هذا الخطأ وتقديم نوع من الإجراء الواجب اتباعه، أو حتى يكفي الاتصال بـ DBA المحلي الخاص بك.

خلاصة

قام هذا الفصل بتقديم أساسيات PL/SQL إلينا كما قام بتزويدنا بالأمثلة الكافية التي تمكننا من البدء في العمل. كما قمنا باستعراض الطرق المختلفة بهدف تكوين إجراءات مخزنة، وآمل أن تتمكن في الحصول على Oracle Procedure Builder لجعل هذه المهمة يسيرة. كما قمنا بتقديم معالجة الأخطاء بتفصيل واف الذي يمكنك من تطوير طريقة أولية قبل قيامك بأي إجراءات. وسنقوم في الفصل التالي ببحث الأحوال التي تتسبب في حدوث الأخطاء.

نأمل أن يكون هذا الفصل قد قدم إليك أسبابا تكفي لإقناعك أن الإجراءات المخزنة هي الطريق الواجب اتباعه وليس شيئا يجب الابتعاد عنه. غير أنه هناك الكثير الذي يمكنك أداءه باستخدام الإجراءات المخزنة، وسيصحبك الفصل القادم خلال المزيد من الأساسيات والعناصر المتقدمة التي تجعل من برامج فيجوال بيسك برامج قوية.

١٠

تكوين الإجراءات المُخزنة

بعد التعرف على الإجراءات المخزنة في الفصل السابق، نحن الآن على استعداد لتعلم طرق الكتابة الصعبة في PL/SQL. وحتى وقت قريب مضى، كان مدى استفادة مطوري فيجوال بيسك من الإجراءات المخزنة محدوداً. وعلى الرغم من استطاعتك تضمين قواعد العمل للتحكم في إدخال البيانات (يجب أن تكون هذه هي طريقتك القياسية في تبديل البيانات)، فإنك لا تستطيع إرجاع البيانات إلى تطبيقك دون معاملات إخراج الإجراءات المخزنة. وقد بدأ هذا يتغير خلال العام الماضي، وربما كان ذلك بتأثير من فيجوال بيسك.

الموضوعات التي سنقوم بمناقشتها هي:

- أنواع بيانات PL/SQL
- تعريف المتغيرات
- جمل وتخصيصات
- جمل التحكم في التدفق
- استخدام جمل SQL في PL/SQL
- خلايا PL/SQL
- أنواع بيانات معرفة بواسطة المستخدم تتضمن السجلات وجداول PL/SQL ومتغيرات الخلايا:
- استخدام الزنادات
- التزويد بتدقيق الحركة
- الإجراءات والفيجوال بيسك

أنواع بيانات PL/SQL

تستطيع أن تحدد داخل إجراء مخزن أو وظيفة واحداً أو أكثر من المتغيرات التي يكون مداها مقصورا على الإجراء بنفس الطريقة التي يتم بها تعريف متغيرات فيجوال بيسك. تقوم بتحديد متغيرات PL/SQL في قسم التعريف في أول الإجراء وهو ما تقوم به عادة في VB.

وقد قمنا بمناقشة أنواع البيانات المتاحة للأعمدة في جداول قواعد البيانات عند تحديد جدول انظر فصل ٦ "تنفيذ قاعدة البيانات"، لكن علينا أن نتعلم هنا مجموعة أخرى. وعلى الرغم من أن بعض الأسماء مازالت كما هي، فإن التحديدات ليست دائماً متطابقة. بالإضافة إلى ذلك، هناك العديد من التحديدات التي توجد فقط في PL/SQL ولا يوجد نظير لها في قاعدة البيانات.

أنواع البيانات المتدرجة

أول وأسهل أنواع البيانات المتاحة لمطوري PL/SQL هو النوع المتدرج. ويتـم إطـلاق كلمـة متدرج على أنواع البيانات البسيطة التي يكون لها قيمة واحدة فقط. وهذا يعني لمطور فيجـوال بيسك أنواعاً مثل الأنواع الصحيحة Integer أو المتسلسلة String. والبديل للأنواع المتدرجـة هي أنواع البيانات المركبة أو المحددة بواسطة المستخدم. سيكون مطور VB على درايـة بـهذه الأنواع لأنها مرادفة للمصفوفات أو الهياكل المحددة بجملة TYPE. يتضمن جدول ١٠-١ أنـواع البيانات المتدرجة المتاحة في PL/SQL.

الجدول ١٠-١: أنواع البيانات المتدرجة في PL/SQL

نوع المتغير	وصفه
BINARY_INTEGER	متغيرات صحيحة على شكل إشارات وتشغل مساحة تخزين أقـل من الأرقام.
BOOLEAN	يقوم بتخزين القيم TRUE وFALSE وNULL.
CHAR(n)	متغيرات تسلسلات حرفية ذات أطوال ثابتة حيـث n هـو عـدد الأحرف التي يمكن أن تصل إلى ٣٢,٧٦٧.
DATE	حقل تاريخ قياسي لأوراكل يحتوي على البيانات والوقت.
LONG	متغيرات تسلسلات حرفية حتى ٣٢,٧٠ بايت. لاحظ الاختلاف عن نوع العمود المناظر الذي يمكنه أن يحتوي على ٢ جيجا بايت.
LONG RAW	متغيرات ثنائية حتى ٣٢,٧٦٠ بايت وهذا أيضا أقل من نوع العمود الذي يمكن أن يحتوي على ٢ جيجا بايت.
NUMBER(n)	متغير صحيح حتى n من الأرقام.
NUMBER(m,n)	قيمة عشرية لعلامة ثابتة مع كون m هو عدد الأرقام قبل العلامـة العشرية وn هي الأرقام بعدها.

الجدول ١٠-١: أنواع البيانات المتدرجة في PL/SQL

نوع المتغير	وصفه
PLS_INTEGER	أرقام صحيحة بشكل إشارات من - ٢١٤٧٤٨٣٦٤٧ إلى + ٢١٧٤٨٣٦٤٧.
RAW	متغيرات ثنائية تخزن حتى ٣٢,٧٦٧ بايت.
VARCHAR2(n)	متغيرات حرفية بطول متغير بحد أقصى حتى العدد n حيث يمكن أن يكون n حتى ٣٢,٧٦٧ بايت.

تستطيع أيضاً استخدام أنواع فرعية عديدة من نوع بيانات NUMBER، وهذه الأنواع الفرعية موضحة في دليل ومرجع مستخدم أوراكل PL/SQL. على أي حال، لا تعطي هذه الأنواع الفرعية ميزة كبيرة للمطور فيما عدا تعريف نوع البيانات الذي كان يفكر فيه المصمم.

تحذير تستطيع أنواع بيانات PL/SQL أن تضم بيانات أكثر من عمود قاعدة البيانات المكتوبة المناظر. على سبيل المثال، يمكن أن يضم VARCHAR2 حتى ٣٢,٧٦٧ بايت في PL/SQL لكنه يمكن أن يضم ٢٠٠٠ بايت فقط في قاعدة البيانات.

متغيرات محددة بواسطة المستخدم (مركبة)

أنواع البيانات المحددة بواسطة المستخدم هي الأنواع التي تقوم بتحديدها أنت في تعليماتك البرمجية وتتضمن أكثر من قيمة (متدرجة) واحدة. على سبيل المثال، يساوي السجل هيكلا محددا بواسطة المستخدم الذي ينشئه مطور فيجوال بيسك بواسطة جملة TYPE. توجد ثلاثة أنواع بيانات محددة بواسطة المستخدم وهي مفصلة في الجدول ١٠-٢. سنتناول كل واحد من هذه الأنواع المعرفة بواسطة المستخدم لاحقا في هذا الفصل لأنها تزود المطور بخصائص يحتاجها جدا.

الجدول ١٠-٢: أنواع البيانات المحددة بواسطة المستخدم في PL/SQL

نوع المتغير	وصفه
Cursor	وهو مؤشر دقيق لسجل في مجموعة نتائج
Record	يسمح لك نوع السجل بتحديد التوليفة الخاصة بك من أنواع البيانات. يشبه هذا النوع هيكلاً في C أو نوع محدد بواسطة المستخدم في VB.
جدول PL/SQL	هذه إضافة حديثة لأنواع البيانات وتعالج قصوراً في PL/SQL. وهي تعمل كجدول قاعدة بيانات على الرغم من أنها تشبه كثيراً تجميع فيجوال بيسك في طريقة العمل. تستطيع أيضاً استخدام هذه كمصفوفات.

تعريف المتغيرات

في PL/SQL، يكون عليك تعريف جميع المتغيرات التي تريد استخدامها في إجرائك. يأتي تعريف المتغيرات في بداية قطعة تعليمات بعد الرأس وقبل جملة BEGIN الأولى. يمكن أيضاً أن يوجد تعريف المتغيرات في قسم التعريف لأي قطعة متداخلة. وكما مع فيجوال بيسك، تقوم أنت بتضمين كل اسم متغير وتتبعه بنوع بياناته. على سبيل المثال، قد ترى تسلسل الأوامر التالي:

```
PROCEDURE test (p_val NUMBER) IS
  v_customer_name       VARCHAR2(50);
  v_customer_number     NUMBER(10);
  BEGIN
  etc.
```

يقوم هذا الجزء من التعليمات بتعريف متغيرين أحدهما نوعه VARCHAR2(50) والآخر نوعه NUMBER(10). تستطيع تعريف جميع المتغيرات التي تريد استخدامها في الإجراء سواء كانت أنواعاً متدرجة مثل NUMBER أو أنواع أكثر تعقيداً وهي التي سنراها في بقية هذا الفصل.

تستطيع أيضاً تحديد قيمة أولية للمتغير في جملة التعريف باستخدام مشغل التخصيص القياسي:

```
v_total_amount    NUMBER (5,2) := 0;
v_data_found      BOOLEAN := FALSE;.
```

معايير تسمية المتغيرات

تتاح العديد من المعايير المقبولة لتسمية المتغيرات في فيجوال بيسك. معظم الأمثلة التي رأيتها لم تحاول تكرار معايير التسمية هذه في إجراء مخزن. ولأن معظمنا سينتج تعليمات PL/SQL أقل كثيراً من تعليمات فيجوال بيسك فإن هذه المعايير ليست شديدة الأهمية.

على أي حال سيكون تدريباً جيداً أن تأتي بمعايير تسمية خاصة ببعض الأجزاء. وعلى الأقل، فإني أقترح التفريق بين معاملات الإدخال (التي أضيف إليها "-P" في أولها) والمتغيرات المعرفة داخلياً (التي أضيفت إليها "-V" في أولها). الفائدة من هذا هي تقليل من احتمال محاولة تخصيص قيمة لمعامل إدخال على الرغم من أن هذا لا يبدو وكخطأ في ترجمة الوقت.

استخدام TYPE% و ROWTYPE%

يكون استخدام أنواع البيانات القياسية في PL/SQL مشابهاً لتلك المستخدمة في فيجوال بيسك. في PL/SQL، يكون لديك الخيار أيضاً لتعريف المتغيرات من نوع مشابه لنوع البيانات موجود في جدول قاعدة بيانات أو من نوع تم تحديده بالفعل. تستطيع فعل ذلك بواسطة سمة TYPE%. يوضح المثال التالي متغيراً جديداً هو movie_length والذي تم تعريفه كنفس النوع بعمود الطول في جدول Movies، ويوضح أيضاً المتغير average_length المعرف كنفس النوع أيضاً.

```
movie_length      movies.length%TYPE;
average_length    movie_length%TYPE;
```

لماذا نستخدم هذا النوع من البيانات؟ حسناً، إذا قررت تغيير دقة عمود جدول من NUMBER(2) إلى NUMBER(5)، لن يكون عليك تغيير كل إجراء مستخدم في هذا العمود. نستطيع ببساطة تحديد متغير من نفس النوع كعمود أياً كان. عندما يتغير العمود، يجب أن يظل

الإجراء يعمل. طبعا، إذا غيرنا أعمدة NUMBER لأعمدة VARCHAR2، فلن يعمل الإجراء في الغالب دون أخطاء تحويل نوع البيانات.

يمكن أن نطبق هذا بشكل أوسع ونقوم بتحديد أنواع المتغيرات المركبة لنفس الهيكل كصف جدول من خلال سمة ROWTYPE% كما في المثال التالي:

```
video_record        videos%ROWTYPE;
```

يحدد هذا النوع متغيرا مركبا هو video_record بنفس هيكل جدول Videos وهو ما يعني أن يحتوي على حقل واحد لكل عمود في جدول Videos. تستطيع الوصول لهذه الحقول بوضع اسم المتغير في أوائل أسماء الحقول. على سبيل المثال، يقوم الأمر.

```
video_record.status
```

بالوصول إلى حقل Status لتسمح لك باستخدامه أو تخصيص قيمة له.

سنرى فائدة هذه العملية لاحقا في هذا الفصل عندما نقوم بمناقشة الخلايا.

تعريف الثوابت

كما يعرف كل مطور، فمن الأفضل استخدام ثوابت بدلا من قيم التعليمات البرمجية الصعبة مع الثوابت الحرفية مثل 3.14159 أو RESERVED خاصة إذا كانت محتملة الاستخدام في اكثر من مكان. قد يكون هذا مجهدا أكثر لكنه يفيد على المدى الطويل لتسهيل الصيانة.

يكون تعريف ثابت في PL/SQL مشابها لتعريف متغير فيما عدا إضافة الكلمة الرئيسية CONSTANT على سبيل المثال، يمكننا تعريف

```
pi          CONSTANT  NUMBER(6,5) := 3.14159;
reserved_rating CONSTANT CHAR(8) := 'RESERVED';
```

تستخدم الثوابت بنفس الطريقة التي تستخدم بها متغيرا فيما عدا أنك لا تستطيع تخصيص قيمة جديدة لها.

ستقوم ببناء إجراءاتك المخزنة بشكل نموذجي في شكل حزمة وتصنيف وتعريفات الثوابت إليها. بهذه الطريقة، ستكون الثوابت متاحة خلال الحزمة.

تلميح استخدام الثوابت هي طريقة نموذجية للتعامل مع رسائل وأرقام الخطأ. تستطيع تنظيم رسالتك الموجودة في الحزمة لتجنب أي إعادة لاستخدام لنفس رقم الخطأ ولتقلل من متاعب الصيانة.

جمل وتخصيصات

تتكون PL/SQL من جمل كأي لغة إجراءات أخرى، حيث تنتهي كل جملـــة بفاصلـــة منقوطـــة وبالتالي تستطيع أن تمتد هذه الجمل لأكثر من سطر من التعليمات البرمجية. إحدى النتائج لـــهذا هي أن هذه الطريقة لتعيين نهاية جملة تسمح لك بتنسيق التعليمات ليكون من السهل قراءتها.

أما التخصيصات فتستلزم المزيد لتعتاد عليها. فبدلا من علامة "=" التي نستخدمها في فيجـوال بيسك يجب أن تستخدم البديل ":=" التي يعتاد عليها مبرمجو باسكال.

```
v_value_1  := 10;
v_complete := FALSE;
v_name     := 'unknown';
v_name_2   := v_name;
```

جمل التحكم في التدفق

بمجرد تعريف متغيراتك ستحتاج لأن تتعود على التحكم في التدفق قبل أن تستطيع إنشاء أكثر من برنامج بسيط. توجد نقطتان أساسيتان مفقودتان في SQL لكنهما ضروريتـــان تمامـــا للبرمجـــة الإجرائية، هما التحكم في تدفق القرارات والحلقات. هاتان النقطتان هما أساس البرمجة الإجرائية وهما أكثر الخصائص أهمية التي توفرها PL/SQL في SQL.

التحكم في تدفق القرارات

جمل التحكم في التدفق التي تحتاجها أولا هي للتحكم في القرارات. ويكون عنصر التحكـــم فـــي القرار للتدفق لتنفيذ جملة برنامج هو جمل IF...THEN يجب أن نراجعها مع مثال لكل منها.

تستخدم جملة IF...THEN لعملية تعيين قرار بسيطة. على سبيل المثال، لتقوم بإعداد متغير هـــو *movie_type* طبقا لمدة طول المادة الفيلمية، استخدم الجمل التالية.

```
movie_type := 'short';
IF movie_length < 120 THEN
  movie_type := 'long';
END IF;
```

IF...THEN...ELSE

تعد الصيغة الأخرى من هذه العبارة البرمجية هي IF...THEN...ELSE وهـي مفيـدة عنـد الاختيار بين ناتجين مختلفين، ومن أحد الأمثلة العملية التي توضح هي القيـام بتعيين رسـالة المتغير بالاعتماد على متغير *movie_rating*.

```
IF movie_rating = 'G' THEN
  message := 'Suitable for children';
ELSE
  message := 'Not suitable for children';
END IF;
```

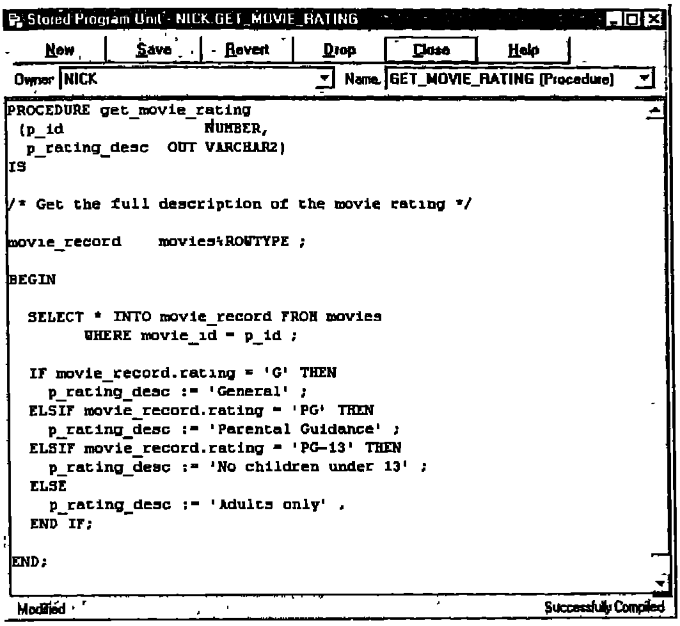
يمكن أن يحتوي تكوين IF...THEN على أي عدد من الجمل بما فيـها جملـة IF...THEN أخرى.

أخيرا، تستخدم أكثر جمل القرارات تعقيدا عندما يكون هناك أكثر من بديلين. يكون هذا أفضـل كثيرا من تداخل عدة جمل IF مع بعضـها. لسـوء الحـظ، لا تقـوم Pl/SQL بتنفيـذ جملـة SELECT...CASE التي نراها في لغات أخرى. يوضح المثال التـالي طريقـة إعـداد متغـير rating_description بالاعتماد على المتغير movie_rating. إذا كانت نتيجة الانتقال واحـدة، فربما كان من الأفضل تخزين الانتقال في جدول تعليمات في قاعدة البيانات.

```
IF movie_rating = 'G' THEN
   rating_description := 'General';
ELSIF movie_rating = 'PG' THEN
   rating_description := 'Parental Guidance';
ELSIF movie_rating = 'PG-13' THEN
   rating_description := 'No children under 13';
ELSE
   rating_description := 'Adults only';
END IF;
```

ملاحظـة تستخدم PL/SQL الصيغة "ELSIF" بدلا من الصيغة الأكثر منطقيـة "elsIf" التي يعتاد عليها مطورو فيجوال بيسك. على أي حـال، سـيقوم المـترجم بتوضيح هذا لك.

يوضح الشكل ١٠-١ إجراءاً كاملاً في باني إجراءات أوراكل الذي يتضمن الكثير من الخصائص التي رأيناها حتى الآن. لتجعل هذا المثال مفيداً، علينا أن نجرب إدخال أمر SELECT...INTO الذي سنناقشه لاحقاً في قسم "استخدام جمل SQL في PL/SQL.




```
PROCEDURE get_movie_rating
 (p_id              NUMBER,
  p_rating_desc  OUT VARCHAR2)
IS

/* Get the full description of the movie rating */

movie_record     movies%ROWTYPE ;

BEGIN

  SELECT * INTO movie_record FROM movies
        WHERE movie_id = p_id ;

  IF movie_record.rating = 'G' THEN
    p_rating_desc := 'General' ;
  ELSIF movie_record.rating = 'PG' THEN
    p_rating_desc := 'Parental Guidance' ;
  ELSIF movie_record.rating = 'PG-13' THEN
    p_rating_desc := 'No children under 13' ;
  ELSE
    p_rating_desc := 'Adults only' ,
  END IF;

END;
```

الشكل ١٠-١: إجراء كامل.

الحلقات

الحلقات هي التكوين الأساسي الثنائي في أي لغة، وتوجد ثلاثة أنواع منها في PL/SQL. يكون استخدام الحلقات مفيداً جداً عند المسح خلال جدول PL/SQL (الذي تمت مناقشته في قسم "جداول وتجميعات PL/SQL) وعند القراءة خلال نتائج خلية منشأة بواسطة جملة SELECT. يعطي هذا القسم أمثلة على الحلقة الأساسية لحساب مجموع المربعات من ١ إلى ١٠.

تركيب LOOP...END LOOP هو عبارة عن حلقة مستمرة تماماً مثل تركيب DO...LOOP في فيجوال بيسك. وهناك ثلاث طرق للخروج من هذه الحلقة هي استخدام جمل EXIT أو EXIT...WHEN أو RETURN. (تخرج RETURN من الإجراء بالكامل).

```
LOOP
sum := sum + (i * i) ;
i := i + 1;
IF i > 10 THEN
  EXIT;
```

```
    END IF;
END LOOP;
```

هناك نسخة بديلة من هذه الحلقة تستخدم جملة EXIT...WHEN. ولها نفس تأثير وضع جملة EXET في قطعة IF، كما في المثال السابق. وهي طريقة أبسط بكثير لإنهاء الحلقة.

```
LOOP
    sum := sum + (i * i) ;
    i := i + 1;
    EXIT WHEN i = 10;
END LOOP;
```

WHILE...LOOP

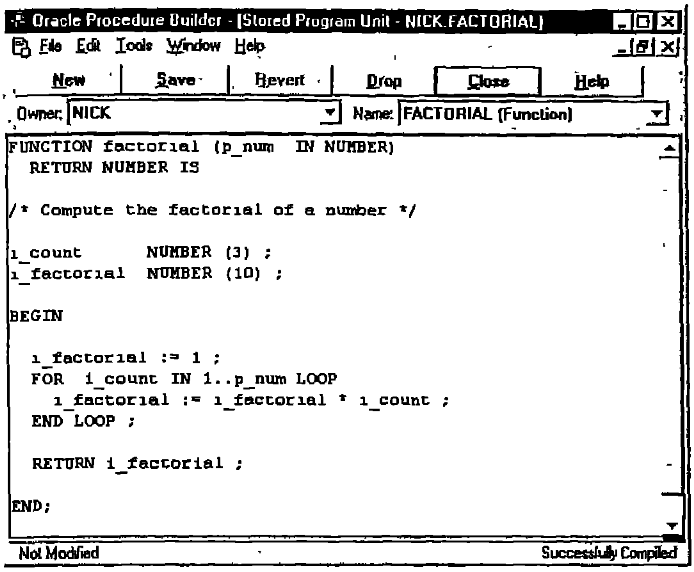
تركيب WHILE...LOOP مشابه جداً لجملة WHILE...DO في فيجوال بيسك وهي تعمل تقريباً بنفس الطريقة في الأمثلة السابقة.

```
i := 1;
sum_square := 0;
WHILE i < 10 LOOP
    sum_square := sum_square + (i * i);
    i := i + 1;
END LOOP;
```

الشكل الأخير من أشكال الحلقات هو FOR...LOOP وهو يشبه حلقة FOR...NEXT في فيجوال بيسك وتعمل هذه الحلقة لنطاق من الأرقام الصحيحة.

```
sum_square := 0;
FOR i IN 1..10 LOOP
   sum_square := sum_square + (i * i);
END LOOP;
```

يتم تحديد مثال لإجراء كامل في الشكل ١٠-٢.



الشكل ١٠-٢ وظيفة بها تركيب حلقة.

تلميح على الرغم من إمكانية تنفيذ الإجراءات لجميع العمليات التي تنفذها الدوال، فإن الدوال لها ميزة إضافية هي إمكانية استخدامها في أي مكان يمكن أن تعمل فيه دالة قياسية لأوراكل (بافتراض أن لديك الامتيازات الصحيحة). أحد الفوائد الكبرى لهذه الخاصية هي إمكانيتها لاستدعاء الدالة من داخل جملة SQL.

استخدام جمل SQL في PL/SQL

لكي نستفيد من PL/SQL، تحتاج أكثر من مجرد جمع المتغيرات واستخدام الحلقات. نحتاج أن نتفاعل مع قاعدة البيانات. تعتبر PL/SQL امتدادا لـ SQL وعلى ذلك نستطيع إدخال أوامر SQL بلا مشاكل في تعليماتك. على أي حال، لا يمكن استخدام جميع أوامر SQL بهذه الطريقة، فهناك فقط نوعين من الأوامر نهتم بها هنا هما النوع الذي يقوم بتحديث البيانات والآخر هو الذي يقوم فقط باسترجاعها.

جمل التحديث

تستطيع استخدام جمل Insert وUpdate وDelete بنفس الطريقة التي تستخدمها بها في جملة SQL عادية كما ناقشنا في فصل ٨، تحديث قاعدة البيانات. توضح التعليمات البرمجية ١٠-١ إجراء قصيرا لتبين كيف تستفيد من جملة Insert.

تعليمات برمجة ١٠-١

```
PROCEDURE movie_insert(
  p_movie_id       IN NUMBER,
  p_name    IN VARCHAR2,
  p_length  IN NUMBER,
  p_rating  IN VARCHAR2,
  p_year    IN NUMBER) IS
/* Insert a new movie into the database */
BEGIN
  INSERT INTO movies VALUES (
    movie_sequence_id.NEXTVAL, p_name, p_length, p_rating, p_year,
    USER, SYSDATE);
  COMMIT;
EXCEPTION
/* Whatever the problem just rollback and
   pass the error back to Visual Basic */
  WHEN OTHERS THEN
    ROLLBACK;
    RAISE;
END movie_insert;
```

تم تصميم الإجراء Movie-Insert لإدراج سجل جديد إلى جدول الأفلام Movies. إضافة جميع البيانات المطلوبة في الإجراء كمعاملات. يستخدم الإجراء INSERT ويتبعه بأمر COMMIT للتأكيد على جعل التغيير نهائيا. تتعامل قطعة EXCEPTION مع أي أخطاء تظهر أثناء حفظ البيانات. وعلى الرغم من ضرورة استخدام ROLLBACK في هذه الحالة لأن الخطأ يشير ضمنا إلى أن السجل الجديد لم يتم إدراجه، فسيكون تدريبا جيدا على تنفيذ جميع الأساسيات.

باستخدام إجراء مخزن، لا يكون على مطور التطبيق أن يكتب أية تعليمات برمجية ليتعامل مع مسلسل. وبالإضافة إلى هذه السهولة في الاستخدام فإن استخدام الإجراء المخزن يؤدي إلى تحسين كفاءة العمل قليلا.

الإجراءات في حيز التنفيذ

لتأمين بياناتك بأكبر قدر ممكن، يجب عليك إنشاء ثلاثة إجراءات منفصلة لكل جدول ليكون للمستخدمين النهائيين إمكانية الوصول إلى واحدة من دوال Insert و Update و Delete.

إذا كانت لديك جداول كثيرة، فقد ترغب في إنشاء ثلاثة إجراءات لكل جدول. على أي حال، يمكن تنفيذ هذه المهمة ببرنامج فيجوال بيسك صغير. سنناقش هذا في فصل ١٤، وحدات الوصول للبيانات.

استخدام SELECT INTO

كما ذكرنا من قبل، لا توجد بأوراكل طريقة مباشرة لإرجاع مجموعة النتائج من جملة SELECT في إجراء مخزن. إذا قمت بإرفاق جملة SELECT بسيطة في تعليمات PL/SQL الخاصة بك، سيظهر لك خطأ ترجمة. على أي حال، تستطيع استخدام جملة SELECT INTO وهي تعديل على SELECT القياسية التي تسمح بقراءة البيانات لمتغير بسيط أو مركب، أو لأكثر من متغير صيغة جملة SELECT هي:

```
SELECT column_list | *
  INTO variable_list | record_variable
  FROM table_list | subquery
  WHERE condition
```

يجب أن تطابق قائمة العمود في جملة Select Into قائمة المتغير. بمعنى أنه يجب أن يكون هناك متغير واحد لكل عمود يقرأ فيه. يجب أن تعطي جملة Select أيضا صفا واحدا فقط وإلا سيظهر خطأ. وكمثال، سنستخدم جملة Select Into لقراءة اسم فيلم نعرف أسمه بالفعل.

```
SELECT movie_name
  INTO v_name
  FROM movies
  WHERE movie_id = p_id;
```

أحد استعلامات جملة Select Into الأخرى هو البحث عن وتخزين رقم المسلسل التالي المتاح الذي يمكننا الوصول إليه من الجدول الثنائي Dual.

```
SELECT movie_sequence.NEXTVAL
  INTO next_sequence
  FROM dual;
```

في الإجراء الذي يقوم بتوليد قيمة رئيسية جديدة في رقم مسلسل، يكون من المفيد إمكان إرجاع رقم المسلسل للوراء للإجراء المستدعي الذي ربما يكون فيجوال بيسك. ستريد استخدام الرقم كإدخال لإجراء آخر لتنشئ علاقة أساسية تفصيلية. على سبيل المثال، تستطيع إدراج فيلم جديد واستخدام القيمة الرئيسية من رقم المسلسل كمعامل إدخال لإدراج صف إلى جدول Videos.

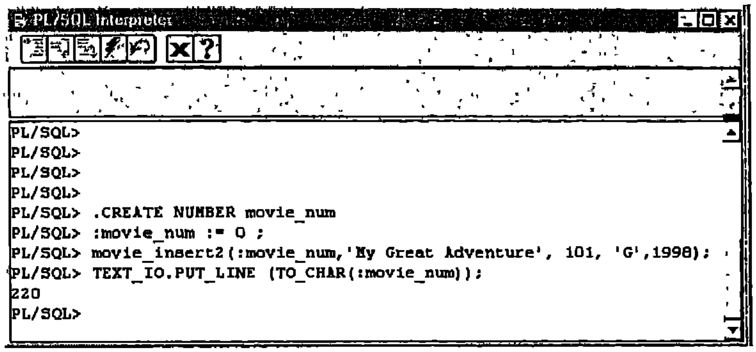
لتسترجع رقم المسلسل، تستطيع تحديد المعامل p_movie_id بطور IN OUT ثم تقوم بإعداد قيمة movie_sequence.NEXTVAL قبل استخدامه كقيمة لجملة Insert "انظر التعليمات البرمجية ١٠-٢". على أي حال، لتنفيذ من هذا من فيجوال بيسك، ستحتاج لاستخدام شيء أكثر تطورا من وحدات الوصول إلى البيانات (DAO) التي لا تستطيع استرجاع معامل إخراج من إجراء مخزن. بدلا من ذلك، سيكون عليك استخدام واحد من الأساليب الأكثر تطورا التالية: وحدات البيانات البعيدة أو وحدات البيانات النشطة أو وحدات أوراكل لـ OLE.

تعليمات برمجة ١٠-٢

```
PROCEDURE movie_insert2(
  p_movie_id IN OUT NUMBER,
  p_name    IN VARCHAR2,
  p_length  IN NUMBER,
  p_rating  IN VARCHAR2,
  p_year    IN NUMBER) IS
/* Insert a new movie into the movie table
   If the formal parameter, p_movie_id is zero it is assigned
   from the next available sequence number
*/
BEGIN
  IF p_movie_id = 0 THEN
    SELECT movie_sequence_id.NEXTVAL INTO p_movie_id
      FROM dual;
  END IF;
  INSERT INTO movies VALUES (
    p_movie_id, p_name, p_length, p_rating, p_year, USER, SYSDATE);
  COMMIT;
EXCEPTION
/* Whatever the problem just rollback and
```

```
   pass the error back to Visual Basic */
  WHEN OTHERS THEN
    ROLLBACK;
    RAISE;
END movie_insert2;
```

تستطيع رؤية تنفيذ هذا في باني إجراءات أوراكل في الشكل ١٠-٣. لاحظ كيف ننشئ متغيرا جديدا هو movie_num وإعداد قيمتها بصفر. نقوم بتنفيذ الإجراء بإدخال قيمة movie_num بالإضافة إلى جميع القيم الجديدة الأخرى. عندما يظهر الإجراء، نتأكد من قيمة movie_num بواسطة الإجراء TEXT_IO.PUT_LINE لنرى قيمة المتغير ٢٢٠. هناك شيئان يجب أن تلاحظهما، هما أنك يجب أن تستخدم النقطتين قبل المتغيرات المرتبطة كما يجب عليك القيمة الرقمية الناتجة إلى حرف بواسطة وظيفة TO_CHAR قبل محاولة عرضها.



شكل ١٠-٣ تنفيذ الإجراء.

إذا كنت لا تستخدم طريقة وصول للبيانات في تعليمات فيجوال بيسك الخاصة بك تسمح لك بقراءة معاملات الإخراج من إجراءات مخزنة، فلا تعتقد أن هناك مشكلة، فلديك بدائل عديدة. تستطيع تنفيذ كل ما تريده من داخل إجراء مخزن واحد كإدراج الصف الأساسي والصفوف التفصيلية أيضا. وعلى الرغم من أن هذا يفيد في بعض المواقف المحددة جدا، فسيكون عليك عادة اللجوء إلى جملة Select من فيجوال بيسك لقراءة الرقم المسلسل التالي المتاح وإرساله إلى جميع الإجراءات المخزنة التي تحتاجه.

يعطي هيكل الإجراء المخزن الموضح في التعليمات البرمجية ١٠-٢ أفضل الخصائص. إذا أدخلت صفر كمعامل p_movie_id، فسيقوم الإجراء بإنشاء قيمة جديدة. وإذا أدخلت قيمة أخرى لا تساوي الصفر فسيقوم الإجراء باستخدامها.

تلميح لا تخف من جعل تعليماتك البرمجية سهلة القراءة وجذابة أيضا. أحد أسهل الأشياء هو تضمين الكثير من الفرغات البيضاء، بمعنى سطور خالية تفصل المناطق المختلفة من التعليمات. تستطيع محاذاة المعاملات أو تعريفات المتغيرات بفراغات أو علامات جدولة. لا يوجد إدارة نسخ في PL?SQL ولذلك فمن الحكمة الاحتفاظ ببيان تشغيل للتغييرات في التعليقات عند بداية الإجراء.

أخطاء في SELECT INTO

في القسم التالي، سنرى المؤشرات وهي مساحات ذاكرة تشير إلى وتسترجع بيانات من مجموعة نتائج يمكن أن تتكون من صف واحد أو أكثر من البيانات أو بدون صفوف من البيانات. تستخدم جملة SELECT INTO ما يسمى بالخلية الضمنية (وتكون ضمنية لأنك لا تحتاج تجديدها لأن ذلك يتم آليا) ويجب أن تنتج صفا واحدا. إذا لم تنتج أية صفوف أو تنتج أكثر من صف، يظهر خطأ ويكون عليك وضع أي وظائف تريد تعيين تعليمات برمجية من قطعة الاستثناء. الأخطاء المحتملة مدرجة بالتفصيل في الجدول ١٠-٣.

تلميح لأن SELECT INTO يجب أن تنتج دائما صفا واحدا فقط في مجموعة نتائجها، فستجد أن استخدام الخلايا الصريحة أكثر مرونة.

الجدول ١٠-٣: الأخطاء التي يمكن أن تنتج من جملة SELECT INTO

الخطأ	وصفه
SELECT INTO	تم إرجاع أكثر من صف واحد بواسطة جملة SELECT INTO.
NO_DATA_FOUND	لم يتم إرجاع أي صفوف بواسطة جملة SELECT INTO.

توضح التعليمات البرمجية ١٠-٣ مثالا لتقييد هذه الأخطاء. نختار عمود v_movie_id. يجب أن تنتج SELECT INTO صفا واحدا فقط. إذا لم ترجع أية صفوف أو أرجعت أكثر من صف، يتم تنشيط قطعة EXCEPTION. تستطيع تتبع أي خطأ معين لتدقيق رسالة الخطأ مثلا.

تعليمات برمجة ١٠-٣

```
FUNCTION get_movie_id (p_movie_name VARCHAR2)
RETURN INTEGER
IS
v_movie_id    NUMBER(10);
BEGIN
SELECT movie_id INTO v_movie_id
FROM movies
WHERE movie_name = p_movie_name;
/* . . . . */
RETURN v_movie_id;
EXCEPTION
WHEN NO_DATA_FOUND THEN
/*  Ignore the NO_DATA_FOUND error (PL/SQL needs some
statement)*/
v_movie_id := 0;
WHEN OTHERS THEN;
/*  Pass other errors back to Visual Basic */
ROLLBACK;
RAISE;
END get_movie_id;
```

يمكن استخدام جملة SELECT INTO لاسترجاع متغيرات عديدة في خطوة واحدة لكنها يمكن أن تستخدم أيضا مع متغير سجل. سنرى مثالا لذلك عندما تنتقل إلى قسم "أنواع بيانات معرفة بواسطة المستخدم" لاحقا في هذا الفصل.

خلايا PL/SQL

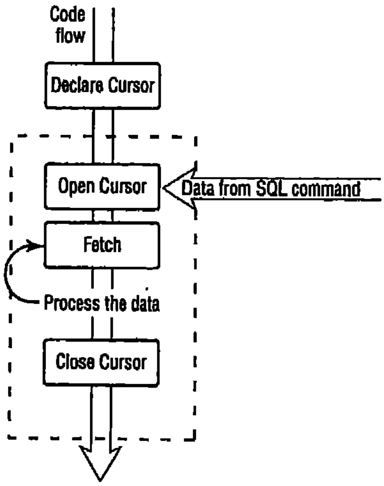
ستتعرض لكلمة Cursor كثيرا في هذا الكتاب، لكنها لسوء الحظ تعني أشياء مختلفة كل مرة. في هذا الفصل، سأستخدم كلمة PL/SQL Cursor أو خلية PL/SQL لأميز بينها وبين client-side أو مؤشرات جانب المزود على الرغم من ورود الاثنين في وثائق أوراكل باسم Cursor. تعتبر خلايا PL/SQL نوعا ثابتا من المؤشرات وهو ما يعني أنك تقوم بتعريفها بجملة SELECT محددة (بطريقة مشابهة لثابت فيجوال بيسك). سنرى لاحقا في هذا الفصل متغيرات الخلايا الأكثر مرونة من خلايا PL/SQL.

كما رأينا في فصل ٢، قاعدة بيانات أوراكل على الذاكرة التي من أهم استخداماتها خاصية المجمع المشترك Shared Pool وهي التي تحجز جمل SQL في مساحة SQL المشتركة Shared SQL Area لكل مستخدم أيضا مساحة SQL خاصة Private SQL Area التي تضم المعلومات الخاصة بالمستخدم. وفي هذه المساحة الخاصة، هناك أيضا تقسيم فرعي يقوم بتخزين نتائج استعلام SQL. تسمح لك خلية PL/SQL بإعداد هذا كمساحة مسماة ثم الوصول إلى المعلومات بداخلها. إذا كنت تعتقد أن هذه المساحة كشبكة بجميع الصفوف والأعمدة التي تم استرجاعها، إذن ستتصرف خلية PL/SQL لمؤشر للصف الحالي. لسوء الحظ، تستطيع فقط التحرك للأمام بالمؤشر، بمعنى أنك تستطيع الانتقال إلى الصف التالي وليس الصف السابق. هذه الخاصية "للأمام فقط" الموجودة في أوراكل هي خاصية أساسية. تتاح القدرة على التمرير خلال مجموعات النتائج التي يعتاد عليها معظم مطوري فيجوال بيسك من خلال مؤشرات المزود، كما سنرى في فصل ١١، تقنية جانب المزود.

هناك خطوات عديدة لاستخدام خلية PL/SQL سنقوم بمناقشتها الآن. يمكن أن ترى تلخيصا سريعا للعملية في الشكل ١٠-٤.

الشكل ١٠-٤

عملية استخدام خلية PL/SQL.



تلميح تكون خلايا PL/SQL ذات فائدة كبيرة في برمجة PL/SQL، ولكن حتى وقت قريب لم يكن بإمكانك بإرجاعها إلى فيجوال بيسك. الطريقة الوحيدة لتفعل ذلك الآن هو أن تقوم بإعداد متغير خلية والوصول إليه بواسطة وحدات أوراكل لـ OLE.

تعريف خلية PL/SQL

قبل أن تستطيع استخدام خلية PL/SQL يجب عليك أولا أن تقوم بتعريفها. وستقوم بـــهذا بشـــكل طبيعي مع التعريفات الأخرى في بداية قطعة التعليمات البرمجية. لتحديد خلية PL/SQL يجـــب عليك أولا أن تقوم بتعريفها. وستقوم بهذا بشكل طبيعي مع التعريفات الأخرى في بدايـــة قطعـــة التعليمات البرمجية. لتحديد خلية PL/SQL، يجب أن نستخدم جملة SELECT.

```
CURSOR video_cursor
  IS SELECT *
    FROM videos
    WHERE status = 'available';
```

تقوم هذه التعليمات بإعداد خلية تسمى video_cursor كمؤشر لهيكل يقابل لقائمة العمود فـي جملة SELECT. تستطيع أيضا تعريف خلية بأكثر من معامل واحد. يسمح لك هذا بفتح الخلية في ظروف مختلفة. وهذا مثال لفتح هذا النوع من الخلايا.

```
CURSOR video_cursor (v_status)
  IS SELECT *
    FROM  videos
    WHERE status = v_status;
```

عندما تقوم بفتح خلية PL/SQL، تدرج قيمة لهذا المعامل وهو المعامل v_status في هـــذه الحالة.

فتح وإغلاق خلية PL/SQL

بمجرد تعريفك لخلية، تحتاج فتح الخلية تماما قبل أن ترى أية بيانات. لم يتم إنشاء مجموعة نتائج حتى الآن. لفتح خلية، نستخدم جملة Open ولإغلاقها نستخدم جملة CLOSE.

```
OPEN video_cursor;
PL/SQL statements . . .
. . .
CLOSE video_cursor;
```

لفتح خلية بمعاملات، تدرج قيم المعاملات في جملة OPEN:

```
OPEN video_cursor('available');
```

ملاحظـــة يجب أن تغلق خلية PL/SQL بأسرع ما يمكن لأن ذاكرة SGA المعينة لـــها لا يمكن إطلاقها حتى تغلق الخلية.

سحب خلية PL/SQL

مع جملة OPEN تكون لدينا الآن مجموعة النتائج في الذاكرة والوسيلة الوحيدة للوصول لها هـــي استخدام جملة FETCH والتي تقوم بسحب الصف التالي من مجموعة النتائج إلى متغيرات داخلية صيغة هذه الجملة هي:

```
FETCH cursor_name | cursor_variable_name
 INTO variable_list | record_name;
```

يفتح المثال الموجود في التعليمات البرمجية ١٠-٤ خلية PL?SQL ويقـــوم بســـحب صـــف البيانات الأول إلى المتغيرات ثم يقوم بإغلاق الخلية. وتكون المتغيرات متاحة للاســـتخدام كمـــا ترغب.

تعليمات برمجة ١٠-٤

```
OPEN video_cursor;
FETCH video_cursor
  INTO v_video_id, v_movie_id, v_status, v_rental_date,
    v_customer_id, v_purchase_date
CLOSE video_cursor;
```

ملاحظـــة لا توجد طريقة للتحرك لأسفل في مجموعة نتائج بواسطة خلية PL/SQL.

حالات الخطأ لخلايا PL/SQL

تحتاج أن تكون على علم ببعض الحالات عند تشغيل خلية PL/SQL. أنت تعلم أنك ستصل إلـــى نهاية مجموعة نتائج عند نقطة معينة. يمكن أن لا تكون هناك أية صفوف في مجموعة النتـــائج. إذا ظهرت هذه الحالة، سيتم الخروج من القطعة الحالية من التعليمات وربما ســـتجد أنـــه مـــن الصعب إنتاج أي برامج مفيدة. بدلا من ذلك، يجب أن تتأكد من عدم حدوث واحدة من حـــالات الخطأ. هذه الحالات متاحة كسمات للخلايا وتم توضيحها في الجدول ١٠-٤.

الجدول ١٠-٤ : سمات خلية بعد عملية سحب.

سمات الخلية	وصفها
%FOUND	تكون هذه السمة مساوية لـــ NULL قبل عملية FETCH الأولى و TRUE إذا كانت عملية FETCH الأخيرة وجدت صفا. وتساوي FALSE إذا لم تجد FETCH صفا.
%ISOPEN	تكون قيمة هذه السمة TRUE إذا كانت خلية PL/SQL أو متغير الخلية مفتوحين وتكون FALSE إذا كانا غير ذلك.
%NOTFOUND	هذه السمة هي عكس سمة FOUND%، بمعنى أنها تكون TRUE إذا فشلت آخر عملية FECTCH في إيجاد صف وتكون FALSE إذا وجدت صفا.
%ROWCOUNT	تحتوي هذه السمة على عدد المرات التي تم فيها سحب صف معين بشكل ناجح. وتكون صفرا قبل عملية FETCH الأولى.

لاستخدام هذه السمات لخلية PL/SQL، نضعها إلى جانب اسم الخلية.

```
video_cursor%NOTFOUND
```

إذا حدثت حالة خطأ خلية لكن تعليماتك تحاول الوصول إلى المتغير الذي تمت قراءته من خلية PL/SQL، سيسبب هذا الفعل خطأ يكون في العادة هو INVALID_CURSOR. لن يكون لديك عندئذ خيار إلا التعامل مع الخطأ في قطعة EXCEPTION.

يتم استخدام سمات خلايا PL/SQL عادة في جمل IF كما يلي:

```
OPEN video_cursor;
LOOP
   FETCH video_cursor INTO v_status;
   EXIT WHEN video_cursor%NOTFOUND;
  . . .
END LOOP;
```

خلايا PL/SQL في حيز التنفيذ

كيف تتناسب خلايا PL/SQL مع هيكل إجراء ما؟ ستقوم باستخدامها لتحصل على مجموعة نتائج من قاعدة البيانات ثم المرور عبر الصفوف واحدا وتعمل من خلالها بأي طريقة تريد. الحقيقة أنك تستطيع تنفيذ أي من هذه المهام في تعليمات فيجوال بيسك الخاصة بك، فلماذا إذن تبحث عن المشاكل بكتابة إجراء بلغة لست معتادا عليها؟

يكون تنفيذ مهامك من خلال إجراء مخزن ذا مميزات كبيرة. فأنت تقلل كثيرا حجم البيانات التي يجب نقلها خلال الشبكة. ستجد أيضا أن معالجة البيانات على نفس الجهاز الموجودة به يكون أكثر كفاءة. بالإضافة إلى ذلك، من الأسهل كثيرا أن معالجة البيانات على نفس الجهاز الموجودة به يكون أكثر كفاءة.

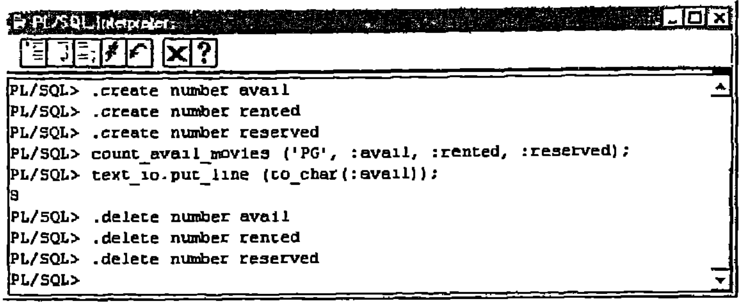
قد ترغب دائما في معالجة مجموعة النتائج بالكامل، ولتفعل ذلك، يجب عليك أن تستخدم الخلية وتعمل مع كل صف على حدة. لعرض هذا التفاعل، سنقوم بإنشاء إجراء لعد أرقام الأفلام في مجموعات حالة متعددة "مثل "rented" أو "available" إلى آخره" حسب تصنيف معين. أحد الطرق لفعل ذلك هو سحب جميع السجلات لفيجوال بيسك وعدها "تذكر أن وظيفة Count ترجع فقط قيمة واحدة لكل جملة SELECT".يسمح لنا استخدام خلية PL/SQL بإنتاج النتائج على الخادم وشحن قيم قليلة فقط تزيد عن معدل الشبكة. توضح التعليمات البرمجية ١٠-٥ كيفية تحديد PL/SQL وفتحها والدخول خلال جميع الصفوف ثم معالجة كل صف على حدة والخروج بعد الصف الأخير.

تعليمات برمجة ١٠-٥

```
PROCEDURE count_avail_movies (
  p_rating   IN VARCHAR2,
  p_avail    OUT NUMBER,
  p_rented   OUT NUMBER,
  p_reserved OUT NUMBER)
IS
/* Count the number of videos in the three categories
  available, rented and reserved, for the specific rating
  that is passed as an input parameter to the procedure */
v_status  VARCHAR2(10); /* input field for the FETCH */
v_avail   NUMBER(5);   /* We need these counters as */
```

```
v_rented   NUMBER(5);   /*  we are using OUT parameters */
v_reserved NUMBER(5);   /*  not IN OUT  */
/* Set the cursor to define the subset of data we want to look at */
CURSOR video_cursor
  IS SELECT status
    FROM videos, movies
    WHERE videos.movie_id = movies.movie_id
      AND rating = p_rating;
BEGIN
  v_avail := 0;
  v_rented := 0;
  v_reserved := 0;
/* Loop through all the rows in the result set that the cursor
   points to and add up the numbers of videos in the various
   categories. */
  OPEN video_cursor;
  LOOP
    FETCH video_cursor INTO v_status;
    EXIT WHEN video_cursor%NOTFOUND;
    IF v_status = 'available' THEN
      v_avail := v_avail + 1;
    ELSIF v_status = 'rented' THEN
      v_rented := v_rented + 1;
    ELSIF v_status = 'reserved' THEN
      v_reserved := v_reserved + 1;
    END IF;
  END LOOP;
  CLOSE video_cursor;
  p_avail  := v_avail;
```

يوضح الشكل ١٠-٥ كيف يمكننا تنفيذ واختبار هذا الإجراء في باني إجراءات أوراكل.



الشكل ١٠-٥

تنفيذ إجراء count_avail_movies.

حلقات خلايا PL/SQL

يمكن لصيغة خاصة من جملة FOR...LOOP أن تجعل العمل مع خلايا PL/SQL أسهل كثيرا. ستقوم آليا بتنفيذ جمل Open وFETCH وCLOSE بدلا منك ثم تدور في حلقات من خلال كل صف متاح. لا تحتاج أنت حتى للتأكد من حالة نهاية الحلقة. وإذا قمنا بتعديل التعليمات البرمجية ١٠-٥ لنستخدم هذا الأسلوب، فسنرى التعليمات البرمجية التالية:

```
...
 FOR video_record IN video_cursor LOOP
   IF v_status = 'available' THEN
     v_avail := v_avail + 1;
   ELSIF v_status = 'rented' THEN
     v_rented := v_rented + 1;
   ELSIF v_status = 'reserved' THEN
     v_reserved := v_reserved + 1;
   END IF;
 END LOOP;
 ...
```

لاحظ أن المتغير Viderd_record يتم إعلانه ضمنيا كسجل من نفس نوع الخلية "في هذه الحالة، يكون هناك متغير واحد".

أنواع بيانات محددة بواسطة المستخدم

يسمى امتداد هام لأنواع المتغيرات التي رأيناها حتى الآن بنوع بيانات محدد بواسطة المستخدم ويعني القدرة على تجميع أكثر من قيمة متدرجة في نوع واحد. ولـ PL/SQL ثلاثة أنواع مختلفة محددة بواسطة المستخدم، وسيؤدي الاعتياد عليها إلى إمكانية مساعدتك ليس فقط

للاستفادة القصوى من كتابة الإجراءات المخزنة ولكنه سيضيف إلى جعبة أدواتك التي تستخدمها كمطور فيجوال بيسك. والأنواع المركبة الثلاثة للبيانات التي سنناقشها هي:

- السجلات
- الجداول أو التجميعات
- متغيرات الخلايا

السجلات

السجل هو نوع بيانات مركب محدد بواسطة المستخدم. وهو عبارة عن تجميع البيانات ذات صلة (حقول) في هيكل محدد. والسجل يكون مطابقا تقريبا لهيكل في فيجوال بيسك تحصل عليه عند استخدامك لجملة TYPE لكنه ربما يكون أكثر فاعلية. في الجزء الأول من هذا الفصل، رأينا كيف نحدد سجلا بواسطة سمة ROWTYPE%. لاستخدام سجل، يجب علينا أولا تعريف نوع البيانات للسجل الجديد في قسم التعريف في الإجراء ثم نقوم بتعريف متغير لهذا النوع.

ملاحظة هذه هي المرة الوحيدة التي يتم استخدام فيها ألفاظ "سجل" و"حقل" في SQL أو في PL/SQL وهما مختلفان عن الصفوف والأعمدة في جدول قاعدة بيانات على الرغم من استخدام السجلات غالبا لتحديد هيكل يمكن قراءته في صف قاعدة بيانات.

لتحديد نوع البيانات الجديد نستخدم جملة TYPE.

```
TYPE record_type IS RECORD (field_list);
```

يحدد record_type نوع المستخدم الجديد الذي يمكن أن تستخدمه عندئذ لتعريف متغيراتك. أما field_list فهي قائمة بأسماء الحقول وأنواع بياناتها. على سبيل المثال، تستطيع تعريف

```
TYPE movie_record IS RECORD (
 movie_id    NUMBER(10),
 movie_name  VARCHAR2(50),
 rating      VARCHAR2(10));
```

وهو هيكل يسمى movie_record ويتكون من ثلاثة حقول هي movie_id, وmovie_name و rating.

تقوم باستخدام هذه الحقول في نفس الظروف كما تفعل في فيجوال بيسك. وهي تجمع سويا متغيرات تكون طبيعيا مجموعة منطقية وتحفظها في هيكل فعلي تستطيع استخدامه خلال

البرنامج. وللإشارة إلى أي جزء من السجل، قم فقط بإرفاق اسم المتغير في أول اسم السجل، قم فقط بإرفاق اسم المتغير في أول اسم السجل مع فصلهما بنقطة.

```
movie_record.movie_id := 1234;
```

السجلات هي طريقة هامة لتسهيل كتابة تعليماتك البرمجية وقراءتها أيضا مما يسهل صيانتها. على أي حال، لا تستطيع إرجاع هيكل سجل إلى تعليمات فيجوال بيسك الخاصة بك.

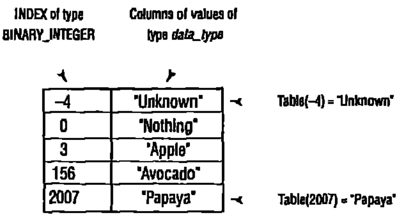
تلميح إذا كنت تستخدم سمة ROWTYPE% عند تحديد سجل، فليس عليك أن تحدد نوع مستخدم، فهو يكون مضمنا عندما تستخدم السمة على سبيل المثال، يقوم movie_record movies%ROWTYPE بتحديد هيكل سجل من نفس النوع كصنف في جدول Movies.

جداول وتجميعات PL/SQL

كانت المصفوفة هي الهيكل المفقود من SQL والإصدرات الأولى لـ PL?SQL. ولا يوجد حتى الآن مصفوفة حقيقية لكنه لا يوجد ما يدعو للقلق لأن لدينا جدول PL?SQL.

لفظ تجميعات هو لفظ أكثر دقة لأنها تعمل بشكل مشابه بالتجميعات التي تراها في فيجوال بيسك ولغات أخرى تعتمد على الوحدات أكثر من المصفوفات المعتادة. ويكون الجدول متخصصا أكثر بالنوع أكثر من تجميع فيجوال بيسك "حيث تستطيع بشكل افتراضي إضافة أي نوع بيانات تقريبا" كما أن الجدول ليس له حدود. أحد طرف عرض جدول PL/SQL تكون كعمودين، يضم العمود الأول الفهرس كمفتاح أساسي بينما يضم العمود الثاني القيمة. يوضح الشكل ١٠-٦ طريقة بناء واستخدام جدول PL/SQL.

الشكل ١٠-٦ طريقة عمل جدول PL/SQL.



ملاحظة وسع أوراكل ٨ مفهوم الجدول لتشمل التجميعات التي تضم الجداول المشار إليها هنا بالإضافة إلى المصفوفات. هذه الأنواع الجدية لبيانات المصروفات لا يمكن الوصول إليها حاليا بواسطة مطوري فيجوال بيسك.

تعريف جدول PL/SQL

لتعريف جدول، يجب علينا تحديد نوع المستخدم مرة أخرى. في هذه الحالة، تكون الصيغة هكذا.

```
TYPE movie_name_type IS TABLE OF VARCHAR2(50)
  INDEX BY BINARY INTEGER;
movie_names        movie_name_type;
```

ثم بعد ذلك تستخدم النوع table-type لتحديد متغيرات بالطريقة العادية.

```
TYPE movie_name_type IS TABLE OF VARCHAR2(50)
  INDEX BY BINARY INTEGER;
movie_names        movie_name_type;
```

لتستخدم هذا في إجرائك، يمكنك الإشارة إليه كما تفعل مع مصفوفة في فيجوال بيسك مع الفهرس الخاص بها.

```
movie_name := movie_names(i);
```

تستخدم PL/SQL هذا الفهرس "في هذه الحالة" للوصول إلى البيانات باستخدامه كمفتاح أساسي للبحث عن الصف في الجدول ثم استخدام عمود القيم للحصول على القيمة. عندما تقوم بتخصيص قيمة لصف مفهرس ليس موجودا حتى الآن، يتم إنشاء صف جديد للقيمة المخصصة. ليس هناك حاجة لزيادة الفهرس في الواقع، يسمح لك الجدول بتمثيل مصفوفة متفرقة بشكل كلف وهي المصفوفة ذات العناصر متفاوتة القيم. تعتبر عناصر الجدول التي لم يتم تخصيصها غير محددة ومن الخطأ محاولة الوصول إلى القيمة.

استخدم جدول PL/SQL

لتوضيح مثال بسيط لاستخدام الجدول، دعنا أولا نملأ جدولا. لجميع الأفلام التي تحتوي على كلمة معينة "انظر التعليمات البرمجية ١٠-٦". يجب علينا أولا تعريف الخلية التي سنستخدمها لاسترجاع مجموعة النتائج. لاحظ الطريقة التي تستخدم بها عبارة WHERE جملة LIKE معقدة. ثم تقوم بتعريف جدول لاستقبال أسماء الأفلام بالإضافة إلى عداد. أخيرا، نقوم بالدخول خلال مجموعة النتائج بتدريج العداد لكل صف جديد ووضع اسم الفيلم في الجدول.

تعليمات برمجة ١٠-٦

```
PROCEDURE get_movies (
  p_word  IN VARCHAR2)
IS
/* Put movies containing a certain word into a a PL/SQL table */
```

```
CURSOR movie_cursor IS
  SELECT movie_name
    FROM movies
    WHERE LOWER(movies.movie_name) LIKE
      '%' || LOWER(p_word) || '%';
TYPE movie_name_type IS TABLE OF VARCHAR2(50)
  INDEX BY BINARY_INTEGER;
movie_names      movie_name_type;
v_count     NUMBER(5);

BEGIN
  v_count := 0;
  OPEN movie_cursor;
  LOOP
    v_count := v_count + 1;
    FETCH movie_cursor INTO movie_names(v_count);
    EXIT WHEN movie_cursor%NOTFOUND;
  END LOOP;
END get_movies;
```

إرجاع جدول PL/SQL إلى الإجراء المستدعي

المشكلة الوحيدة في التعليمات البرمجية ١٠-٦ هي أنها لا تحقق أي شيء! للاستفادة من الجدول، تحتاج لإرجاعه للإجراء المستدعي. دعنا نفترض أنك تريد إرجاع جدول من إجراء مخزن بـه أسماء الممثلين المشتركين على الأقل في فيلمين (انظر التعليمات البرمجية ١٠-٧). ما هو مـهم من وجهة نظر فيجوال بيسك هو أننا يجب أن نعين للإجراء معاملاً يحتوي على أكبر عدد مـن السجلات التي سنسمح بها. بمعنى آخر، ما هو أكبر عدد من القيم نريد استقباله مرة أخرى مـن الإجراء.

ملاحظـة تعتبر جداول PL/SQL هي أكثر الأساليب تطورا التي ستصادفك عندما تجري اتصالا بين أوراكل وفيجوال بيسك ولذلك فهي تستحق الجهد الـذي تبذله لإتقانها. لاسترجاع جدول إلى فيجوال بيسك، سيكون عليك استخدام وحدات البيانات البعيدة (RDO) أو وحدات البيانـات النشـطة (ADO) أو وحدات أوراكل OLE (0040). سنناقش الأساليب الأنسب في الفصول ذات الصلة.

لإدخال جدول لإجراء المعامل، نحتاج لتحديد نوع البيانات الجديد في حزمـــة. وهـــذه هـــي الطريقة الوحيدة لكي يتعرف أوراكل النوع المحدد بواسطة المستخدم قبل تنفيذ الإجراء. تكـــون العملية مشابهة لجملة TEPE في فيجوال بيسك التي تقوم فيها بتحديد نوع متغير قبل إدخال متغير من هذا النوع في برنامج فرعي. يتم توضيح جزئي الحزمة "المحتويات والهيكل" في التعليمـــات البرمجية ١٠-٧.

تعليمات برمجة ١٠-٧

مكونات الحزمة

```
PACKAGE movie_package IS
/* Define the table type */
TYPE char_table_type IS TABLE OF VARCHAR2(50)
  INDEX BY BINARY_INTEGER;
/* Define the procedures in the package body */
PROCEDURE get_stars (
  p_table_size       NUMBER,
  p_stars        OUT char_table_type);
/* any other package definitions */
END movie_package;
```

هيكل الحزمة

```
PACKAGE BODY movie_package IS
PROCEDURE get_stars (
  p_table_size       NUMBER,
  p_stars        OUT char_table_type)
IS
  /* Return a PL/SQL table with all the stars who have
     been in at least two movies.
     The table will only hold a certain number of
     entries, up to the limit defined by p_table_size. */
  CURSOR star_cursor IS
    SELECT first_name || ` ` || last_name
      FROM participants, movie_participants
```

```
    WHERE participants.participant_id =
          movie_participants.participant_id
     AND role = 'A'
    ORDER BY last_name;
  v_name            VARCHAR2(50) := '';
  v_last_name        VARCHAR2(50) := '';
  v_num_found        NUMBER(3) := 0;
  v_table_row        NUMBER(2) := 0;
BEGIN
   OPEN star_cursor;
   LOOP
 FETCH star_cursor
   INTO v_name;
     EXIT WHEN star_cursor%NOTFOUND OR
       v_table_row >= p_table_size;
     IF v_name = v_last_name THEN
       v_num_found := v_num_found + 1;
     ELSE
       v_num_found := 1;
     END IF;
     IF v_num_found = 2 THEN
       v_table_row := v_table_row + 1;
       p_stars (v_table_row) := v_last_name;
     END IF;
     v_last_name := v_name;
   END LOOP;
   CLOSE star_cursor;
END get_stars;
END;
```

التعليمات الواردة في التعليمات البرمجية ١٠-٧ هي واحدة من أكثر الأمثلة تعقيدا في هذا الفصل وتحتاج إلى تدقيق أكثر. يحدد الجزء الأول محتويات الحزمة حيث يتم نوع البيانات المحددة بواسطة المستخدم char_table_type لجدول متغير VARCHAR2(50). ومع تحديد الجدول، يمكن استخدامه كمعامل للإجراء. تجد مكونات الحزمة الإجراء get_stars أيضا.

الجزء الثاني من الحزمة هو الهيكل الذي يتضمن أية إجراءات ووظائف. تذكر أن رأس الإجراء المحدد في هيكل الحزمة يجب أن يتطابق تماما مع الرأس في مكونات الحزمة. يحدد الإجراء get_stars أولا خلية تختار أسماء الممثلين من قائمة المشاركين وترتيبهم بالاسم. ويقوم بفتح الخلية والتنقل خلالها ليحسب كل صف بالترتيب حتى يتم الوصول إما لنهاية مجموعة النتائج أو الحد الأقصى للجدول، أيا كان الأول. تتم المعالجة بسهولة فهي تلاحظ تكرار أثنين أو أكثر من السجلات المتتالية لنفس الممثل وتضمينها إلى الجدول.

الحلقات الضمنية للجداول

يمكن تكامل الجداول بشكل صحيح مع الحلقة الضمنية لخلية لتبسيط تعليماتك تماما

```
. v_video_row := 0;
FOR movie_rec IN
  (SELECT movie_name FROM movies ORDER BY movie_name) LOOP
  v_video_row := v_video_row +1;
  video_table(v_video_row) := movie_rec.movie_name;
END LOOP;
```

في هذا المثال، تنفذ جملة FOR...IN...LOOP أولا عملية SELECT لإنشاء مجموعة نتائج. ثم تقوم بإنشاء متغير حلقة هو movie_rec الذي تقوم بتحديده آليا كسجل من نفس نوع مجموعة النتائج (في هذه الحالة، عمود واحد فقط). ثم تقوم بالدخول في حلقة خلال التعليمات بتدريج العداد v_video_row ثم تخصيص Movie_Name للصف الحالي لجدول Video_Table. وكما ترى، يمكن تحقيق الكثير في أسطر قليلة من التعليمات.

تحذير عندما تصل إلى إدخال جدول، تأكد من أنه محدد سابقا. تأكد من أنك قمت بتخصيص قيمة للفهرس الذي استخدمته قبل أن تجرب استخدامه في أي مكان آخر. إذا لم تفعل، سيثير الوصول إلى بيانات الجدول خطأ. بعض الإصدارات من PL/SQL لم تتعامل مع الخطأ بشكل جيد وهو ما يجعل تصحيح الأخطاء شاقا جدا.

متغيرات الخلايا

أخيرا وصلنا إلى أحدث وأقوى أنواع البيانات المحددة بواسطة المستخدم وهي متغيرات الخلايا. ظهرت هذه المتغيرات لأول مرة في الإصدارة ٢,٣ من PL/SQL مع أوراكل ٧,٣. وهي تشترك

مع خلايا PL/SQL في أشياء كثيرة، ولكن عند تعريف هذه الخلايا في استعلام يتم تخصيص قيمة فقط لمتغيرات الخلايا (بمعنى أنها مرتبطة باستعلام) في طور التشغيل. تتم المقارنة في وثائق أوراكل بين خلية PL/SQL وثابت وبين متغير خلية ومتغير. ولكل مكانه، لكن تقديم متغيرات الخلايا ربما يحل محل استخدام خلايا PL/SQL لأن المتغيرات ليست لها أية عيوب.

أحد مميزات الخلية هو مرونته في أنك لا تستطيع فقط تخصيص جملة SELECT له في طور لتشغيل ولكنك تستطيع إعادة تخصيصها في أي وقت. لكن الميزة الأهم ربما تكون أنه بتحديد متغير الخلية كمعامل إجراء سليم فإنك تستطيع إرجاع نتائج من إجراء مخزن. ما تقوم بإدخاله في حقيقة الأمر هو مؤشر إلى مجموعة النتائج وليس مجموعة النتائج نفسها. تفيد متغيرات الخلايا جدا لإدخال مجموعات نتائج بين إجراءات PL/SQL، لكن من الممكن الآن استخدامها من فيجوال بيسك.

متغيرات الخلايا مع جداول PL/SQL هي من أكثر الطرق تعقيدا للتعامل مع أوراكل. على أي حال، ففي معظم المواقف التي ستصادفها، ستستخدم متغيرات الخلايا بشكل أساسي لإرجاع مجموعة نتائج إلى إجراء الاستدعاء الخاص بك ببساطة وسهولة. لن تكون هناك معالجة، لأنك لا تستطيع تغيير القيم في مجموعة النتائج.

إن وظيفة الخلايا هي محاكاة جملة SELECT من إجراء مخزن وبالتالي تشبه مميزاتها العروض. تقوم أنت بإخفاء تعقيدات الاستعلام والمستخدم من الوصول لجميع الأعمدة لجدول قاعدة. لا يستطيع المستخدم النهائي أن يكتشف أي جدول أو مجموعة من الجداول تزود بالبيانات. وبتأخير تخصيص تسلسل الاستعلام حتى وقت التشغيل، يمكنك جعل مجموعة النتائج معتمدة على اسم المستخدم أو التاريخ أو أي شيء ترغبه، وهي أداة أخرى فعالة تضيفها إلى أدواتك.

ملاحظة قبل أن تبتهج كثيرا، يجب أن تلاحظ أن استرجاع متغير خلية في برنامج فيجوال بيسك يحتاج منك لاستخدام وحدات أوراكل لـ OLE (الإصدارة ٢,١ أو الإصدارات الأحدث). ليس متغير الخلية جزءا من محتويات ODBC وربما لن يكون أبدا لأنه يعيد مؤشرا لا مجموعة نتائج. الجانب المشرق هو أن إصدار موزعي OLE/DB يمكن أن يتيح هذه الإمكانية لنا بشكل أفضل.

```
TYPE movie_cv_type AS REF CURSOR RETURNS movies%ROWTYPE;
```

يبدأ التعريف بالطريقة العادية بالكلمة الرئيسية TYPE واسم نوع متغير الخلية الذي نريد تحديده، في هذه الحالة يكون movie_cv_type. الجزء التالي "AS REF CURSOR" يعني أن

هذا النوع الجديد يشير إلى خلية. أخيرا، يعرف الجـزء "RETURNS movies%ROWTYPE" نوع البيانات الذي ستشير الخلية إليه. هذا الجزء يكون اختيارا، فإذا تركته (بكتابة غير مركــزة weak typing) ؟؟؟ فإنه يوفر مرونة أكبر مع احتمال إثارة خطـ. وربما يكون أقل كفــاءة مــن الكتابة المركزة ؟؟؟ الموضحة بالأعلى. في المثال، نقوم بتعريف أن الخلية ستشير إلـــى هيكــل محدد بنوع الصف لجدول MOVIES.

نستطيع بعد ذلك تعريف متغير خلية لهذا النوع المحدد بواسطة المستخدم.

```
movie_cv    movie_cv_type;
```

فتح متغير خلية

لأن الاستعلام الذي يستخدمه متغير خلية لا يكون محددا سابقا، فيجب تحديده عندما تفتحه. بـــدلا من جملة OPEN، نستخدم الآن جملة OPEN...FOR. صيغة هذه الجملة هي:

```
OPEN cv_name FOR select_query;
```

وهي تقوم بفتح متغير الخلية cv_name وتحديد مجموعة نتائج هي select_query.

```
OPEN movie_cv FOR
  SELECT * FROM movies
    WHERE year_of_release<1990;
```

لاحظ أنه يجب عليك التأكد من ان نوع الهيكل الذي يشير إليه متغير الخلية هو نفسه هيكــل نتيجة الخلايا التي قمنا بتخصيصها له بواسطة OPEN. في هذه الحالة، قمنا بتأكيد هذا بالإشــارة بمتغير الخلية لصف من جدول MovIes واستخدام "* SELECT" في الاستعلام وذلك لنعلم أنهما سيتطابقان. إذا أردنا اختيار بعض الأعمدة فقط، فسيكون علينا تحديد هيكل نموذج RECORD من هذا النوع ثم نشير بمتغير الخلية إليه.

سحب السجلات

على الرغم من أن الفائدة الرئيسية لمتغيرات الخلايا لمطوري فيجوال بيسك هي إرجاع مجموعــة نتائج لفيجوال بيسك، فإن متغيرات الخلايا تفيد أيضا كنسخة أكثر مرونة مــن خلايــا PL/SQL. يمكن معالجتها تقريبا بنفس الطريقة بواسطة أمر FETCH في الحلقة. على سبيل المثال، فسيعمل الأمر.

```
FETCH movie_cv INTO movie_rec;
```

تماما لكلا النوعين من الخلايا.

إغلاق متغير خلية

عندما تغلق متغير خلية، لا تظل مجموعة النتائج محددة. من المفيد إغلاق أي مجموعة نتائج تماما عندما تكون غير مستخدمة، لكن هذا يحدث آليا عند انتهاء الإجراء إلا إذا قمت بتعريفها كمعامل بواسطة سطور IN OUT>

```
CLOSE movie_cv;
```

تمرير متغيرات الخلايا

ربما كان أهم ما يميز متغيرات الخلايا هي أنها يمكن أن تمرر مجموعات نتائج للخلف إلى برنامج فيجوال بيسك الخاص بك، لكن كيف لنا أن ننفذ هذا بدقة إذا خرجت المتغيرات عن النطاق "أصبحت غير محددة" عندما ينتهي الإجراء؟ تكمن الإجابة في تحديد متغير الخلية كمعامل صحيح للإجراء بواسطة طور IN OUT.

ملاحظة تستطيع تحديد متغيرات الخلايا كمعاملات طور IN، لكن هذا لا يفيد كثيرا إلا إذا كنت تقوم بالتمرير بين إجراءات PL/SQL.

لتمرير أية أنواع بيانات محددة بواسطة المستخدم إلى إجراء، لا بد أن تكون جزءا من حزمة. ستقوم الحزمة بتعريف نوع متغير الخلية. تعطي التعليمات البرمجية ١٠-٨ مثالا لإجراء في حزمة يمرر متغير خلية إلى الخلف لإجراء استدعاء. يفتح هيكل الإجراء متغير الخلية بواسطة جملة SELECT. أيا كانت الصفوف التي يتم اختيارها فسيتم إرجاعها إلى الخلية.

تعليمات برمجة ١٠-٨

مكونات الحزمة

```
PACKAGE movie_package IS
/* Define the cursor variable type */
TYPE movie_cv_type IS REF CURSOR RETURN movies%ROWTYPE;
/* Define the procedures in the package body */
PROCEDURE get_old_movies (
  p_year     NUMBER,
  old_movies IN OUT movie_cv_type);
/* Add any other package definitions here */
```

```
END movie_package;
```

هيكل الحزمة

```
PACKAGE BODY movie_package IS
PROCEDURE get_old_movies (
  p_year      NUMBER,
  old_movies IN OUT movie_cv_type) IS
  /* Return a cursor variable with all the records
     released prior to the passed year */
  BEGIN
    OPEN old_movies FOR
      SELECT * FROM movies
        WHERE year_of_release < p_year
        ORDER BY year_of_release;
  END get_old_movies;
  END;
```

نقوم فقط بفتح متغير الخلية بواسطة جملة SELECT، ولا تقوم بإغلاقه لأنه سيصبح عندئذ غير معرف وغير ذي فائدة لبرنامج فيجوال بيسك. هذا مثال بسيط تستطيع الحصول على ما هو أفضل منه بكثير. تستطيع استخدام متغيرات الخلايا بنفس طريقة استخدام خلايا PL/SQL، لكن متغير الخلية هو الطريقة الوحيدة لإرجاع مجموعة نتائج من إجراء مخزن إلى برنامج فرعي للاستدعاء.

حالات الخطأ

المشاكل التي يمكن أن تصادفها مع متغيرات الخلية هي نفسها مشاكل خلايا PL/SQL الثابتة. أرجع إلى قسم "حالات الخطأ لخلايا PL/SQL" لمزيد من التفاصيل.

استخدام الزنادات

قمنا بتخصيص معظم هذا الفصل على الإجراءات على حساب الزنادات التي تشبه معالجات الأحداث في فيجوال بيسك عند حدوث فعل مسجل محدد، يتم تنفيذ الزناد دون تحكم من المستخدم. يجب أن تحافظ على الزنادات بسيطة وقصيرة قدر الإمكان. سيجد مطور فيجوال بيسك أن أوضح تطبيق للزنادات هو القدرة على تطبيق معلومات التدقيق على الصفوف، لكنها

أيضا تستخدم في كل شيء ابتداء من البرامج الفرعية للمراجعة من البيانات وحتى دعم أوراكل نفسه لتكرار قاعدة البيانات.

كمثال على استخدام زناد، تخيل أن لدينا مجموعة سجلات قابلة للتحديث في فيجوال بيسك. "سترى تفاصيل أكثر عن هذا في فصل ١٤ "وحدات الوصول للبيانات". إذا كانت لديك معلومات التدقيق على صفوف وأعمدة جدولك مثل such as Time_Stamp وUser_Name فأنت لا تريد أن يزودك تطبيق VB بها. إذا قمت بإعداد على الجدول، تستطيع عندئذ تزويد قيم الأعمدة هذه آليا.

لإعداد زناد تحكم تدقيق بواسطة باني إجراءات أوراكل، نفذ الخطوات التالية:

١- من وحدات قاعدة البيانات Database Objects الموجودة في مستكشف الوحدات Object Navigator، اختر Tables.

٢- انقر الجدول الذي تريد إنشاء الزناد عليه (تذكر أن هذه الزنادات مرتبطة بالجداول)، واختر Triggers في الشكل ١٠-٧، اخترنا جدول Movies.

٣- انقر أيقونة New على الجانب الأيسر من المستكشف وسترى مربع حوار لإدخال معلومات الزناد (راجع الشكل ١٠-٧).

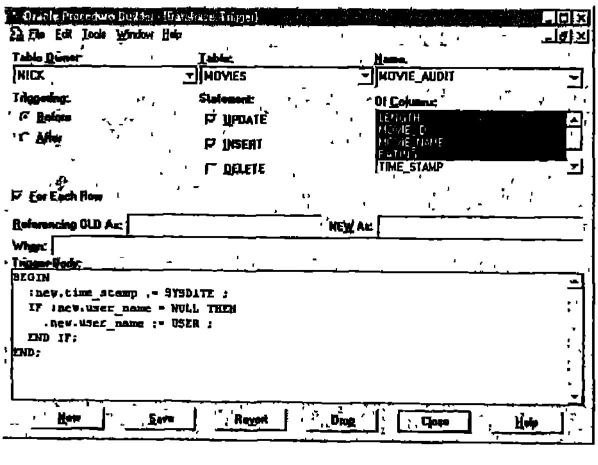
٤- انقر زر New الموجود بأسفل مربع الحوار.

٥- انقر زر Be Fore لخيار Triggering واختر Update وInsert لخيار Statement. ولـ Update يجب أن تختار جميع الأعمدة التي تريد تنفيذ زناد عليها.

٦- تأكد من تعيين خيار "For Each Row".

٧- أعط الزناد اسما مناسبا مثل "movie-audit".

٨- أملأ TrIgger Body كما هو موضح في التعليمات البرمجية ١٠-٩.



الشكل ١٠-٧: شاشة TrIgger في باني الإجراءات.

تعليمات برمجة ١٠-٩

```
BEGIN
:new.time_stamp := SYSDATE;
IF :new.user_name = NULL THEN
  :new.user_name := USER;
END IF;
END;
```

عندما يقوم مستخدم بإدراج صف جديد أو يقوم بتحديث واحد من أعمدتك المختارة، سيتم تشغيل الزناد. ويؤكد تشغيل هذا الزناد أن هناك طابع وقت صحيح واسم مستخدم. هناك عبارة IF لعمود User-Name التي تقوم بالتحديث إذا كانت القيمة الداخلة تساوي صفر NULL. تسمح لك هذه العبارة بتحديد القيمة في تعليمات فيجوال بيسك البرمجية ولكنك لا ترغب في تزويد هذه المعلومة.

لاحظ أن بإمكانك نظريا أن تستخدم نفس الأسلوب لتوفر مفتاحا أساسيا من مولد مسلسل إذا كان هناك مفتاح مفقود. وعلى الرغم من أن هذا يبدو وكأنه يربط رقم المسلسل بالجدول، فإن هذا الأسلوب كان هناك مفتاح مفقود. وعلى الرغم من أن هذا يبدو وكأنه رقم المسلسل بالجدول فان هذا الأسلوب له بعض الحدود الكبيرة لمطوري فيجول بيسك تعتمد خلايا ODBC لمجموعات السجلات القابلة للتحديث على مفتاح، والخلايا تحتاج للوصول إلى المفتاح للتحديث ولن تعطي أي نتائج مع القيمة صفرNULL تستطيع أن تحدث تغييرا كبيرا بتعيين المفتاح لقيمة معينة مثل الصفر وتحوله إلى مفتاح صحيح في الزناد على أي حال ستخرج مجموعة سجلات من التزامن مع قاعدة البيانات وسيكون عليك القيام بالتحديث. يؤدي هذا إلى فائدة مجموعة السجلات القابلة للتحديث. في فصل ١٨، وحدات أوراكل لـOLE، سترى الطريقة الوحيدة لاستخدام هذا الاسلوب بشكل صحيح من فيجوال بيسك.

ملاحظة تتاح قيمة حقل العداد الجديدة في أكسس بمجرد تنفيذك لطريقة ADD NEW، لكن هذه الخاصية غير متاحة مع جداول أوراكل.

التزويد بتدقيق حركة

أخيرا، سنرى طريقة لاستخدام أساليب الإجراءات المخزنة لتخزين أرشيف بالأحداث التي قام المستخدمون بتنفيذها. في فصل ٦ "تنفيذ قاعدة البيانات"، تعاملنا مع معايير الجداول وفوائد إضافة

طابع وقت واسم مستخدم للجدول لمعرفة من قام بآخر تغييرات في صف البيانات. يجب عليك أن تفكر في إضافة هذا المستوى الأساسي للتتبع لأي جدول يحتوي على بيانات حركة. قد لا ترغب في إضافتها إلى جداول التعليمات.

أحد التحسينات الواضحة هي إضافة جدول لتخزين مسار تدقيق لأي إجراء مخزن تم تنفيذه بشكل صحيح. وهذه الخطوة تكون سهلة نسبيا. يجب أولا أن تقوم بإعداد الجدول الذي سيقوم بتخزين المعلومات:

```
CREATE TABLE procedure_audit (
CREATE TABLE procedure_audit (
  audit_id    NUMBER(10),
  procedure_name  VARCHAR2(50),
  procedure_parameters VARCHAR2(300),
  time_stamp        DATE,
  user_id             VARCHAR2(20));
```

والآن في كل إجراء، نستطيع إضافة جملة لحفظ جميع المعلومات ذات الصلة بالجدول كما هو موضح في التعليمات البرمجية ١٠-١٠.

تعليمات برمجة ١٠-١٠:

```
PROCEDURE my_proc (
  P_param1          NUMBER,
  p_param2 VARCHAR2,
  p_date     DATE) AS
  v_1  varchar2(80);
BEGIN
/*  . . . */
/*  PL/SQL statements */
/*  . . . */
/* Add the procedure audit trail */
  INSERT INTO procedure_audit VALUES (
    audit_sequence.NEXTVAL,
    'my_proc',
    TO_CHAR(p_param1) || ', '' ||
    p_param2 ||
```

```
   ''',TO_DATE(''' || TO_CHAR(p_Date,'DD-MON-YYYY') ||
   ''',''DD-MON-YYYY''),',
SYSDATE, USER);
 COMMIT;
EXCEPTION
/* Whatever the problem just rollback and pass
   the error back to Visual Basic */
  WHEN OTHERS THEN
    ROLLBACK;
    RAISE;
END my_proc;
```

ليس هذا الإجراء معقدا كما يبدو من أول وهلة. نحن نقوم بإدراج سجل جديد في جدول التدقيق ولذلك نحتاج تزويد الحقول الأربعة بقيم. يأتي المعرف من رقم المسلسل التالي المتاح مع audit_sequence.NEXTVAL ويأتي اسم الإجراء تاليا ويجب أن يتم ضبطه حسب كل إجراء تكتبه. تقوم.

أنت بالتزويد بتسلسل يحتوي على جميع المعاملات بنفس الصيغة التي تستخدمها لاستدعاء الإجراء.سيسبب لك هذا أصعب المشاكل لأن من الصعب التأكد من وجود الصيغة الصحيحة تماما لديك. والقواعد الواجب اتباعها هي كما يلي:

- يمكن للقيم الرقمية أن تظهر كما هي فيما عدا القيمة الأولى التي يجب أن يتم تحويلها إلى قيمة حرفية.
- يجب إحاطة القيم الحرفية بعلامات اقتباس مفردة.
- يجب تحويل التواريخ أولا إلى تسلسل حرفي حتى تعلم تماما الصيغة الموجودة لديك، ويجب أن توضع هذه التواريخ في وظيفة TO-DATE كما يجب أن تظهر في تسلسل المعاملات كما تفعل في تعليماتك في VB.

```
. . . ,TO_DATE('21-OCT-1998','DD-MON-YYYY'), . . .
```

- يجب أن يتبع متغير بفاصلة.
- أخيرا، يجب تضمين طابع الوقت ومعرف المستخدم، وهي ما توفره وظائف أوراكل الإضافية ولذلك فهي سهلة الاستخدام.

ماذا تقدم هذه العملية لنا؟ إنها تعطينا جدولا بسجل تدقيق لكل إجراء قمنا بتطبيقه فيه. يكون لدينا مسار لكل إجراء تم استدعاؤه وأي معاملات تم تمريرها ومن قام بذلك ومتى.

ملاحظة لأن هذا الأسلوب يهدر الوقت، فقد تم تضمينه في برنامج مثال عينة فيجوال بيسك في فصل ١٤ "وحدات الوصول للبيانات".

بالإضافة إلى توفير مسار تدقيق، يسمح الأرشيف أيضا بإمكانية إعادة تكوين البيانات إذا تمت إزالتها بشكل غير مقصود. يجب أن يكون لدينا نظام لاختيار جميع الحركات المدققة ذات الصلة ثم إعادة تنفيذها ببساطة واحدا واحدا بترتيبها الأصلي. أخيرا، يسمح الأرشيف لك بتكرار هذه المعلومات في مكان آخر لأوراكل أو حتى في قاعدة بيانات مختلفة.

ملاحظة إذا كان تطبيقك يسمح للمستخدمين بتحديث الصفوف بدلا من مجرد إدراجها وحذفها "تتبع بعض تطبيقات الحركة هذا النظام"، إذن يجب أن تفكر في أن يكون لديك أربعة أعمدة تدقيق. يجب أن تحدد هذه الأعمدة المستخدم الذي قام بإنشاء الصف ووقت تنفيذ ذلك بالإضافة إلى المستخدم والوقت مرة أخرى عند تحديث الصف لآخر مرة.

الإجراءات وفيجوال بيسك

رأينا الأساليب المتنوعة التي تستطيع استخدامها عند إنشاء إجراءات مخزنة في PL/SQL، ولكن إلى أي مدى تفيد هذه الأساليب مطوري فيجوال بيسك؟ الطريقة الوحيدة لتمرير بيانات من إجراء مخزن إلى أي إجراء آخر، سواء داخليا إلى إجراء أو وظيفة أخرى "PL/SQL" أو خارجيا لبرنامج مبني في فيجوال بيسك، تكون عن طريق المعاملات. لا ترجع "PL/SQL" مجموعة نتائج بنفس طريقة خادم مايكروسوفت SQL وSybase.

في فصول لاحقة، سنناقش الطرق المتنوعة للوصول للبيانات بالإجراءات المخزنة، لكن تلخيص الخصائص الموضح في الجدول ١٠-٦ سيعطيك مقدمة لذلك.

الجدول ١٠-٦: الوصول للإجراءات المخزنة من فيجوال بيسك.

طريقة الوصول للبيانات	إمكانيتها
عناصر التحكم في البيانات	ليست ممكنة
وحدات الوصول للبيانات (DAO)	تستطيع استخدام معاملات طور IN المتدرجة فقط.
وحدات البيانات البعيدة (RDO)	تستطيع استخدام معاملات طور OUT وIN المتدرجة وجداول PL/SQL مع أحدث المشغلات.
وحدات بيانات (ADO) ActIvex	تستطيع استخدام معاملات IN وOUT المتدرجة وجداول PL/SQL مع أحدث المشغلات
وحدات أوراكل لـ OLE (0040)	تستطيع استخدام IN وOUT المتدرجة وجداول PL/SQL ومتغيرات الخلايا.

لأن هذه المنطقة لاتصال أوراكل وفيجوال بيسك قد تغيرت كثيرا على مدى العام الماضي، يجب أن تتأكد أن لديك على الأقل الإصدار أوراكل ٧,٣ مع إما أحدث مشغلات ODBC أو إصدارة من 0040. من المحتمل أن تتم إضافة خصائص جديدة في العم القادم أو نحوه.

خلاصة

ناقش هذا الفصل جميع نواحي إجراءات PL?SQL التي يحتاج معرفتها مبرمجو فيجوال بيسك. بعد مراجعة أنواع البيانات والتعريفات، تعرضنا للتحكم في التدفق مع تكوينات IF...THEN والحلقات. تم تطوير عمليات PL/SQL. هذه هي الخصائص التي تسمح لك بإرجاع أنواع بيانات معقدة إلى تطبيق فيجوال بيسك المستدعي الخاص بك. أوضح الفصل استخداما نموذجيا للزنادات ثم وفر مثالا لكيفية التدقيق في إجراء مخزن. ثم أنهينا الفصل بنظرة على خصائص PL/SQL المتاحة حاليا من طرق الوصول البيانات التي سنراها في النصف الثاني من هذا الكتاب.

الجزء الثالث

تقنية الوصول للبيانات

- تقنية جانب المزود
- الوصول لأوراكل من خلال فيجوال بيسك
- موضوعات عن التصميم باستخدام فيجوال بيسك
- وحدات الوصول إلى البيانات

١١

تقنية جانب المزود

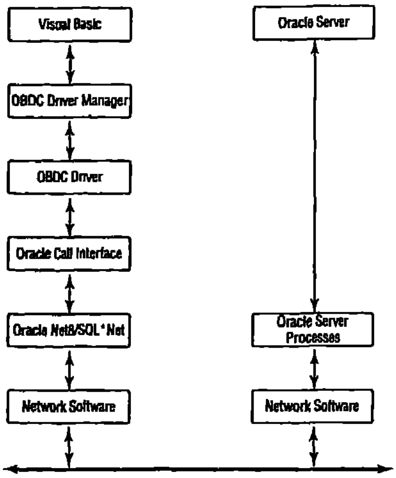
من أصعب الأشياء على الفهم معرفة كيف يتصل برنامج فيجول بيسك الخاص بك بأوراكل. وخاصة فيما يتعلق بالشبكات كإعداد Net8 وSQL*NET وما تقدمه ODBC لمستخدمي أوراكل. سيستكشف هذا الفصل مكونات البرامج المتعلقة بنقل أوامر SQL ومجموعات النتائج عبر الشبكة بالإضافة إلى كيفية إعداد هذه العملية لتعمل بسهولة. ستكون الموضوعات التي سنناقشها هي:

- طبقات البرامج
- برامج الشبكات
- معيار ODBC
- الوصول للبيانات حول العالم وOLE-DB
- الخلايا
- الوصلات
- طرق الوصول البديلة

طبقات البرامج

أولاً، سنراجع ما يحدث بالتحديد عندما تريد الاتصال بأوراكل من برنامج فيجوال بيسك خاص بنا. إذا نظرت إلى الشكل ١١-١، سترى الكثير من الطبقات من البرامج المرتبطة بالعملية. في هذا الفصل، سنرى إعداد العمليات التي تحدث على جهاز المزود وكيفية تحدث على جهاز المزود وكيفية إعدادها. في هذه المرحلة بالتحديد، تكون ODBC كمعيار واقعي هي أهم واجهة تطبيق بين قواعد بيانات فيجوال وSQL مثل أوراكل. على أي حال، مع فيجوال بيسك ٦، بدأنا التوجه نحو تقنية جديدة تسمى OLE-DB التي ستزيد أهميتها باطراد. بدأت OLE-DB كواجهة تطبيق بين VB ومشغلات ODBC لكن واجهات تطبيق OLE-DB الأساسية أصبحت متاحة بالتدريج أكثر وأكثر.

طبقات البرامج المستخدمة بين فيجوال بيسك وأوراكل يتم تفصيلها في الأقسام القليلة التالية.



شكل ١١-١

طبقات البرامج المرتبطة بفيجوال بيسك أوراكل.

مدير مشغلات ODBC

مدير مشغلات ODBC هو الطبقة الأولى من البرامج التي سيصل إليها إذا تم استخدام طريقة ODBC. يتم توفير المدير Manager في الملف oDBC32.DLL. يقوم مدير المشغلات بتحميل مكتبة الربط الديناميكي الصحيحة لمشغل ODBC على الذاكرة. إذا كنت تستخدم أسماء مصادر البيانات (DSNS)، سيجد مدير المشغلات مشغل ODBC الصحيح من الإدخالات في التسجيل Registry، ثم سيستدعي بعض البرامج الفرعية للتهيئة لإعداد المشغل. عندما تقوم بتنفيذ أوامر SQL، سيوجه مدير المشغلات الاستدعاء لنقاط الإدخال الصحيحة في مشغل ODBC بالإضافة إلى تدقيق بعض المعاملات.

مشغل ODBC

مشغلات ODBC هي منتجات برامجية تشكل رابطا خاصا بقواعد بين التطبيق وقاعدة البيانات. ويتم تحميلها والتحكم فيها بواسطة مدير مشغلات ODBC وفي حالة مشغل أوراكل ٣٢ بين أوراكل (؟؟؟) توجد في ملف يسمى SQO32-XX.DLL تكون فيه xx هي إصدارة قاعدة البيانات مثل ٧٣ أو ٨٠. يوجد مشغل ODBC الذي يكون مع فيجوال بيسك ٦ في الملف MSORCL32.DLL. تنفذ مشغلات ODBC استدعاءات وظيفة ODBC بمستوى بسيط وهي مسئولة عن نقل أوامر SQL من التطبيق إلى قاعدة البيانات ونقل النتائج ونقل النتائج مرة أخرى من قاعدة البيانات إلى التطبيق.

تتضمن وظائف مشغل ODBC ما يلي:

- تأسيس الوصلة
- إدارة المزود – الخلية
- نقل الأوامر وصيغ البيانات لتناسب قاعدة بيانات معينة
- إدارة الخطأ – التعليمات

يجب أن يتصل المشغل بأي طبقات برامج خاصة بقواعد البيانات تكون ضرورية وهي في حالة أوراكل تشمل واجهة استدعاء أوراكل وبرامج الشبكات وإما Net8 لأوراكل٨ أو SQL*Net للإصدارات السابقة من قاعدة البيانات.

طبقة OLE-DB

توفر طبقة OLE-DE واجهة تطبيق متطورة لقاعدة بيانات أوراكل بالإضافة إلى قواعد بيانات أخرى. سنناقش هذه الطبقة بالتفصيل لاحقاً في هذا الفصل. فمع ODBC قمنا باستخدام مشغل ODBC ومدير المشغلات، ومع OLE-DB نستخدم مزود بيانات. يمكن لمطوري أوراكل أن يستخدموا أسلوبين: واحداً مع مزود بيانات ODBC الذي يقع بين مشغلات ODBC الموجودة (MSDASQL.DLL)، وآخر هو مشغل أساسي يتجاوز ODBC ويتصل مباشرة بطبقة واجهة تطبيق الاستدعاء لأوراكل (في ملف MSDAORA.DLL).

واجهة تطبيق الاستدعاء لأوراكل

واجهة تطبيق الاستدعاء لأوراكل (OCI) هي واجهة تطبيق برمجة تطبيقات (API) للوصول إلى خادم أوراكل. وهي تأتي على نفس القرص المدمج مع برامج أوراكل للشبكات وتوجد في الملف ORAXX.DLL. وهي تشبه كثيراً ODBC API إلا أن OCI خاصة ليس فقط بأوراكل ولكن أيضاً بلغة البرمجة التي يتم استدعاؤه منها. OCI هي واجهة تطبيق أساسية لأوراكل سواء كنت تستخدم ODBC أو OLE-DB أو وحدات أوراكل OLE أو حزمة أخرى.

وOCI هي API كاملة الخصائص قادرة على أداء جميع الوظائف الموجودة في خادم أوراكل بما فيها تحليل وتنفيذ جمل SQL والتحكم في الحركة وتنفيذ جمل PL/SQL. وتماماً مثل ODBC API، يتم تنفيذها بمستوى بسيط.تقريبا ثلاث دزينات من استدعاء وظائف API وهي بيئة معقدة للعمل بها.

ملاحظة تؤدي OCI دوراً مشابهاً لمكتبة قاعدة البيانات المستخدمة مع خادم SQL.

Net8 وSQL*Net

Net8 وSQL*Net هي طبقات البرامج على جهاز المزود التي تسمح بالوصول من المزود للخادم ومن الخادم للمزود لقواعد بيانات أوراكل ٨ وأوراكل ٧ على التوالي عبر أي شبكة. وكمطور فيجوال بيسك، ستكون ملماً بهذه المنتجات كمتطلب لتوفير طريق من تطبيق فيجوال بيسك الخاص بك إلى الشبكة بواسطة طبقة OCI. على أي حال، توفر الطبقات أيضاً طريق اتصال لتطبيقات شبكات أخرى Network Listener. سنرى المزيد من الطبقات في قسم "إعداد Net8 وSQL*Net".

برامج الشبكات

عندما تقوم بإعداد Net8 أو SQL*net، يتم توجيهك لبروتوكول الشبكة الذي ستستخدمه، ويتم عندئذ تركيب برامج الشبكات المناسبة على جهاز المزود. الهدف من سياسة أوراكل للشبكات هو بناء طبقة نقل مشتركة ومتغيرة الخواص لتشغيل البرامج عليها. والنتيجة هي أن خادم أوراكل وبرامج المزود تستطيع العمل مع أي بروتوكول شبكة وعلى حاسبات ابتداءً من الحاسبات الشخصية إلى شبكات حاسبات IBM.

تتكون برامج الشبكات من طبقات عديدة:

- واجهة تطبيق شبكة شفافة (TNS) وهي أساس لبرتوكول الشبكة المشترك وهي مستقلة عن أي مجموعة برتوكولات شبكة أساسية.
- مهيأ برتوكول أوراكل وهي طبقة برامج خاصة بالحاسب تقع بين الـ TNS وبروتوكول الشبكة الأساسية.
- مجموعة بروتوكلات خاصة بالشبكات وهي برتوكول أساسي مثل TCP/IP وDECNET وSPX/IX.
- يوجد نفس هيكل الشبكة عند جميع المزودين والخوادم، وقد تقوم بتشغيل نفس الشبكة، مع تركيب صغير أو ربما تقوم بتشغيل برتوكولات مختلفة تؤدي نفس المهمة بشكل صحيح كالراوتر والجسور.

إجراء التنصت

يعمل إجراء التنصت على خادم قاعدة البيانات وينتظر طلبات الوصلات من تطبيقات كبرنامج فيجوال بيسك الخاص بك. قمنا بتقديم هذه الخاصية في فصل ٢، بنية قاعدة بيانات أوراكل وكوناها في فصل ٣، تكوين قاعدة بيانات. إذا كنت تعمل بواسطة الخادم المتعدد سلاسل العمليات على يونيكس أو Open VMS، سيقوم إجراء التنصت بترتيب تنفيذ وصلتك من خلال إجراء الموزعات Dispatcher. الخادم المتعدد سلاسل العمليات هو وسيلة فعالة للتعامل مع عدد كبير من طلبات الوصلات. على أي حال، يجب أن تعلم أنه قد يكون من الصعب تركيبه والعمل عليه بشكل جيد.

إجراءات الخادم

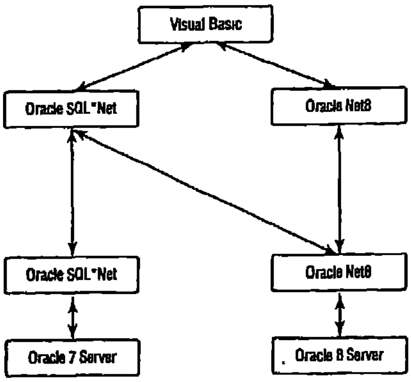
صادفتنا إجراءات الخادم أيضا في فصل ٢ الذي قمنا فيه بتوصيفها كإجراءات جانب الخادم التي يتم إعدادها لكل وصلة مستخدم "لطور الخادم المكرس" أو لمجموعة من الوصلات "لطور الخادم المتعدد سلاسل العمليات".

إعداد Net8 وSQL*NET

بافتراض إعداد برنامج التنصت على الخادم "راجع فصل ٣، تكوين قاعدة بيانات"، فسنكون جاهزين لإعداد جهاز المزود للاتصال به. تحدث جميع الاتصالات إما من خلال OC أو من خلال برنامج أوراكل للشبكة الذي يسمى Net8 لأوراكل٨ و SQL*Net للإصدارت السابقة من خادم أوراكل NET8 هو ثاني أكبر إصدار بعد SQL*Net ٢,٣. كما سترى في الشكل ١١-٢ على الخادم، تحتاج Net8 للوصول إلى قاعدة بيانات لأوراكل٨ وSQL*Net للوصول إلى قاعدة بيانات لأوراكل٧ على الرغم من إمكانية متنصت NET8 وصل المستخدمين بقاعدة بيانات أوراكل٧. أما بالنسبة للمزودين، فلديك حرية أكبر. تستطيع بشكل عام استخدام SQL*Net أو Net8 للاتصال بأوراكل٧ أو أوراكل٨. وبسبب هذا الموقف، فمن الأنسب تماما استخدام SQL*Net على أجهزة المزودين حتى يأتي الوقت الذي يكون فيه مزايا فنية لاستخدام Net8 لمطور فيجول بيسك. عند تلك النقطة، سيكون عليك تقريرإذا كنت ستضع نوعي البرامج على مزوديك "يمكن أن يتواجد معا بسلام" أو إذا كنت ستحدث كل شيء إلى أوراكل٨ وNet8. ولا فرق إذا اخترت العمل بواسطة Net8. أو الاستمرار مع SQL*NET، فالتقنيان متشابهان جدا. تستطيع تركيبها باختيار المنتج من قائمة برنامج تركيب أوراكل Oracle Install.

شكل ١٦-٢

استخدام Net8 وSQL*NET.



لإعداد الشبكة، تحتاج نسخ الملف tnsnames.ora من خادمك إلى مزودك. إذا كنت تشغل ويندوز ٩٥، فستنسخ هذا الملف إلى C:\ORAWIN95\NET8\ADMIN مع SQL*Net أو C:\ORAWIN95\NET8\ADMIN مع NET8. إذا كنت تشغل مزود ويندوز NT، فستخدم الدلائل المقابلة Orant لملفك tnsnames.ora. وهذا كل ما يجب عليك عمله. على أي حال، عليك أن تتأكد أن جميع الخوادم التي تريد الوصول إليها محددة في ملف tnsnames.ora.إذا لم تكن الخوادم محددة، فيجب عليك أن ترجع إلى برنامج Easy Config لتقوم بتحديدها قبل أن تقوم بنسخها إلى حاسباتك الشخصية "راجع فصل ٣، تكوين قاعدة بيانات".

تلميح على الرغم من أن مؤسسة أوراكل لا تحبذ أن تقوم بتحرير ملف tnsnames.ora يدويا، فعلي أن أعترف أنه على جهاز المزود وفي طور التطوير أقوم بتحرير الملف بشكل مستمر في Notepad. إذا كنت تجيد العمل مع Notepad وتصنع تغييرات صغيرة، فإن التحرير يكون مناسبا جدا. على أي حال، بمجرد أن تقوم بذلك فلن تستطيع الرجوع مرة أخرى إلى برنامج Easy Confing لأنه لن يتعرف على صيغة ملفك بعد الآن. نقطة أخيرة جديرة بالملاحظة هنا هي أنك إذا قمت بتحرير ملفات tnsnames.ora أو "listener.ora" بنفسك، فلا ترفق أية حروف جدولة لن SQL*Net أو Net8 لن يعملا بشكل جيد.

إنشاء وصلة

عندما تتداخل الأشياء، ستجد أنك تحتاج فهم كيفية معرفة Net8 أو SQL*Net لمكان إنشاء الوصلة. لهذه العملية، يجب أن تفهم استخدام ملف tnsnames.ora. يفيد فهم هذا الملف فيما يخص إعداد اسم مصدر بيانات ODBC الذي سنناقشه لاحقاً في قسم "إعداد مصدر بيانات ODBC" توضح التعليمات البرمجية ١١-١ صيغة ملف tnsnames.ora.

تعليمات برمجية ١١-١

```
Listing 11.1
Service_name
  (DESCRIPTOR=
  (ADDRESS_LIST=
     (ADDRESS=
       (COMMUNITY=community)
       (PROTOCOL=protocol)
       (protocol specific Information)
      )
    )
    (CONNECT_DATA=
      (SID=sid)
    (GLOBAL_NAME=global_database_name)
    )
  )
```

يجب أن تتوافر عدة قيم في ملف Tnsnames.ara:

Service-name اسم الخدمة التي تريد أن تعطيها لوصلة قاعدة البيانات هذه. يسمى أيضا الخدمة ببديل قاعدة البيانات database alias. إذا استخدمت واحدا من برامج Easy Confing لإعداده، فسيكون هذا هو الاسم الذي ستحصل عليه.

ملاحظة يسبب اسم الخدمة قدرا من الارتباك. يظن الكثير من المطورين أنـــه اســـم قاعدة البيانات، لكن هذا ليس ضروريا. لسوء الحظ، فإن كلا مـــن دخـــول Oracle و ODBC يستخدم العديد من المصطلحات مثل اسم قاعدة البيانـــات database name وسلسلة الوصـــل connect string وقـــاعدة بيانـــات database وخدمة service إلى آخره. المصطلـــح database alias أو بديل قاعدة البيانات هو الأقرب والأكثر دلالة على المعنى.

Community: اسم الجماعة يحدد جزءا مختارا من الشـــبكة يجـــب أن يســـتخدم نفـــس البروتوكول للشبكة باستمرار لكن اختيار الأجزاء المختلفة يرجع إليك أنت. إذا كان لديـــك شبكة واحدة فقط، فتستطيع الاحتفاظ فقط باسم جماعة واحد واستخدام الاسم الافتراضي وهـــو World.

Protocol: وهو بروتوكول الشبكة المستخدم في البحث عن خادم قاعدة البيانـــات، ربمـــا يكون "TXP" على سبيل المثال.

protocol-specific information: سترى لكل بروتوكول مختلف معلومات مختلفـة مطلوبة لإيجاد الخادم مع TCP/IP، يجب أن توفر معلومات المنفذ Port المضيف Host.

SID: وهو محدد النظام لقاعدة بيانات أوراكل. وهو القيمة التي تقوم بإعدادها في متغـــير بيئة Oracle-SID في ويندوز NT، أو في ORA-SID على حاســـابات UNIX و Open VMS. يجب أن يحدد SID بشكل فريد قاعدة بياناتك على اشبكة.

global_database_name: وهذا اسم تستطيع استخدامه لتمييز قاعدة البيانات فـــي بيئة متعددة قاعدة البيانات. تحصل على الاسم باستخدام المعـــامل db_name فـــي ملـــف التهيئة باسم الجماعة ويفصل بينهما نقطة، "Movies.World" على سبيل المثال. إذا كـــان لديك قاعدة بيانات واحدة فقط، تستطيع تجاهل هذا الحقل دون مشاكل.

كمثال لكيفية توصيف الخدمة بشكل واقعي، توضح التعليمات البرمجية ١١-٢ جزءا مـــن ملـــف tnsnames.ora الذي رأيناه في فصل ٣. وهي توضح اسم خدمة orc1.world. في هذه الحالـــة، لكي يجد المتنصت قاعدة البيانات على الشبكة، فإنه يحتاج استخدام بروتوكول TCP/IP والبحث عـــن مضيف يسمى SCCNT01 "المحدد في الملف hosts" بإجراء التنصت علـــى منفـذ ١٥٢١. محـــدد النظام "SID" الذي يريد الاتصال به هو ORC1.

تعليمات برمجة ١١-٢

```
Listing 11.2
orc1.world =
 (DESCRIPTION =
  (ADDRESS =
      (PROTOCOL = TCP)
      (HOST = SCCNT01)
      (PORT = 1521)
  )
  (CONNECT_DATA = (SID = ORC1)
  )
 )
```

ومع توفر هذه المعلومات، دعنا الآن نلقي نظرة على كيفية تأسيس قاعدة البيانات الصحيحة عندما تطلب وصلة من برنامج فيجوال بيسك الخاص بك.

١- تقوم بطلب وصلة من برنامج فيجول بيسك الذي يحدد اسم خدمة. سيكون هذا غالبا عن طريق اسم مصدر بيانات ODBC أو OLE-DB لكنه ربما يكون إشارة لاسم الخدمة إذا كنت تستخدم وحدات أوراكل (OLE).

٢- يبحث Net8 أو SQL*Net في ملف tnsnames.ora عن اسم الخدمة (ORC1.World).

٣- يجد Net8 أو SQL*Net عنوان الشبكة في مكان تنفيذ إجراء التنصت. مثل SCCNTO1 على منفذ ١٥٢١. يتم البحث عن اسم SCCNTO1 في ملف المزود hosts لإيجاد عنوان IP المشابهة أو ربما يتم حله بواسطة خادم DNS.

٤- يتم تمرير طلب الوصلة لإجراء التنصت لإعداد الوصلة.

٥- بافتراض أن المتنصت تم إعداده مع SID المطلوب من SID-LIST في ملف listener.ora وبافتراض تشغيل المتنصت. يتم إنشاء الوصلة بقاعدة البيانات.

ملاحظـــة المنفذ المحدد الذي سيتم إنشاء الوصلة الفعلية عليه. يتم تخصيص منفذ جديد لهذا الغرض.

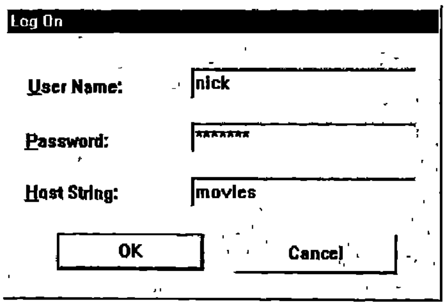
اختبار إعدادك للشبكة

عندما تقوم بإعداد جهاز مزود ليصبح جاهزا للاتصال بأوراكل. يجب أن تسلك طريقة منتظمة لاختبار كل شيء. يجب أن يكون أول ما تقوم به في يومك هو اختبار كفاءة اتصال الخادم للتأكد أولا من اتصال المتنصت بملف listener.ora ثم بملف tnsnames.ora يعمل بشكل صحي. بمجرد التأكد من ذلك ونسخك ملف tnsnames.ora للمزود، عند ذلك فقط يجب عليك اختبار الوصلة من الحاسب الشخصي لمزودك.

أسهل طريقة لاختبار وصلتك على المزود والخادم تكون بواسطة برنامج SQL*Plus. عندما تشغل هذا البرنامج، يجب أن ترى شاشة الدخول المعروفة كما في الشكل ١١-٣. أملأ التفاصيل لاسم المستخدم الخاص بك وكلمة السر وسلسلة الوصلة. وهذا الحقل الأخير هو اسم الخدمة "بديل قاعدة البيانات" الذي يحدد قاعدة البيانات التي تريد الاتصال بها. يجب أن يكون هذا الاسم هو اسم الخدمة وهو واحد من الأسماء التي قمت بإعدادها في ملف tnsnames.ora.

شكل ١١-٣

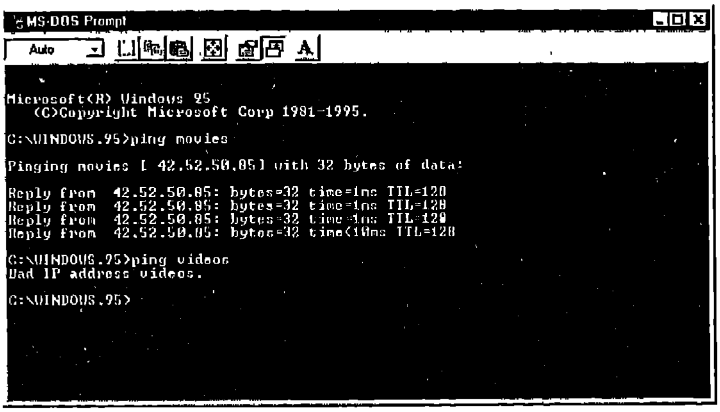
مربع حوار دخول SQL*Plus المعروف.



مساحات مشاكل محتملة

إذا أعددت برامج الشبكة الخاصة بك بشكل صحيح، فلن يكون لديك أية مشاكل. على أي حال، ستتسلل بعض الأخطاء الآدمية. وإذا لم تستطع الاتصال بتطبيق SQL*Plus، يجب عليك مراجعة الأسباب المحتملة:

- تأكد من تشغيل قاعدة البيانات.
- تأكد من تشغيل المتنصت شغل أمر STATUS في المتنصت وتأكد من انتظاره لطلبات وصلات على بديل قاعدة بياناتك (راجع فصل ٣).
- تأكد من إعداد اسم الخادم في ملف hosts على جهاز مزودك. ينظم هذا الملف عناوين الشبكة لاسم الخادم. على سبيل المثال، ففي عالم TCP/IP، سيتكون من إدخالات توضح عناوين IP لاسم الخادم مثل movies 42.52.50.85.
- إذا تم إعداد اسم الخادم، حاول تتبعه ببرنامج PING. يقوم هذا البرنامج بإرسال طلب إلى الخادم ليتأكد من أنه متصل. يوضح الشكل ١١-٤ مثالا تنفيذيا على ذلك للبحث عن خادم يسمى Movies على شبكة TCP/IP. إذا لم تحصل على رد، فيجب عليك عندئذ استشارة مسئول شبكتك.
- هناك خطأ آخر يمكن أن يحدث عندما تعدل ملف tnsnames.ora بنفسك. إذا كان الملف يحتوي على حروف جدولة فلن يكون مزود SQL*Net/Net8 قادرا على قراءته.



الشكل ١١-٤
استخدام برنامج PING لاكتشاف خادم

إمكانيات تتابعية

لأوراكل منظورا في إنشاء قاعدة البيانات ذات الإمكانيات العالية. تتضمن الخيارات تشغيل قاعدة البيانات في طور Hot Standby وتشغيل خادم أوراكل المتوازي (OPS) أو أحد خيارات استنساخ أوراكل المتطورة. والنتيجة هي إمكانية الحصول على خادم آخر إذا تعطل الخادم الحالي. مهمة إعداد وإدارة هذه البيئات هي وظيفة متقدمة تتعدى مجال هذا الكتاب، لكنك قد تعمل في نظام يكون قد أعد

بهذه الطريقة. ولتستفيد من قواعد بيانات أخرى يجب أن يتعرف تطبيق فيجوال بيسك الخـــاص بــك على مكان قاعدة البيانات البديلة. تقوم بفعل هذا في ملف tnsnames.ora.

يتم تحديد tnsnames.ora بواسطة ADDRESS-LIST بعنوان خاص بقاعدة بياناتك كما رأيــت في التعليمات البرمجية ١١-٢. يمكن أن يكون لديك ADDRESS-LIST به العديد من العناوين كمـــا هو موضح في التعليمات البرمجية ١١-٣. عندما يحاول تطبيق فيجوال بيسك الاتصـــال بـــأوراكل. فسيحاول Net8 أو SQL*Net العنوان الول وإذا لم تستطيع الاتصال معه سيحاول الثاني ثم الثالث وهكذا.

تعليمات برمجة ٣-٢

```
Listing 11.3
Orc1.world =
 (DESCRIPTION =
   (ADDRESS =
        (PROTOCOL = TCP)
        (HOST = sccnt01)
        (PORT = 1521)
   )
   (ADDRESS =
        (PROTOCOL = TCP)
        (HOST = sccnt02)
        (PORT = 1526)
   )
   (CONNECT_DATA=
     (SID=ORC1)
     (GLOBAL_NAME=ORC1.WORLD)
   )
 )
```

على الرغم من أن تحديد عدة عناوين يبدو نموذجيا فإن هناك خللا ضئيلا قد يؤدي إلـــى مشـــكلة

كبرى عند التنفيذ. عندما يترك Net8 الوصلة لإجراء التنصت فإنه يتجاوز باعتبار أن مهمته قد انتهت. إذا كان إجراء التنصت موجودا وعاملا ولكن أوراكل لا يعمل، يتم عندئذ فقد طلب الوصلة. يجب أن تتأكد من إغلاق المتنصت قبل أن تبدأ في قاعدة البيانات التالية. للتحايل على هذه المشكلة، يجب أن تضيف بعض الخطوات مع استعمال العناوين المتعددة. عندما يقوم تطبيق فيجوال بيسك بإنشاء الوصلة، يجب أن يختبرها بجملة SELECT بسيطة. (اعتقد أن SELECT * FROM GLOBAL_NAME مناسبة تماما لأنها تسمح لك بمعرفة ما إذا كنت متصلا بالإضافة إلى تحديد قاعدة البيانات التي تتصل بها). إذا فشلت هذه الوصلة الآن أو في أي وقت آخر، يجب أن تكون مستعدا للاتصال بقاعدة البيانات البديلة.

تلميح في يونيكس "مع cron" وOpenVMS "بتسليم عمل"، تستطيع إعداد إجراء لمراقبة قاعدة البيانات دوريا واستدعاء إجراء التنصت إذا تعطلت طبعة أوراكل.

ODBC

ODBC هو الأسلوب الأقوى حاليا للاتصال بين فيجوال بيسك وأنظمة إدارة قواعد البيانات العلاقية البعيدة، مثل أوراكل. أصبح ODBC هو المعيار العام لمعظم مزودي قواعد البيانات الذين يدعمونه تماما. في الواقع، فإن عدم القدرة على إنتاج مشغل ODBC يمكن أن يكون له أثر كبير على القيمة المالية لأي منتج.

بالإضافة إلى توفير الاتصال بقواعد البيانات العلاقية، فإن هناك واجهات تطبيق ODBC متاحة لنطاق كبير من تطبيقات أخرى من تطبيقات سطح المكتب مثل مايكروسوفت أكسس وأكسل "وهو المتوقع" وحتى التطبيقات التي لا ترفقها عادة بقواعد البيانات، مثل Visio.

هناك بعض التطبيقات التي تكون مصادر بياناتها خاصة بها وصيغة بياناتها ليست قريبة أبدا من قاعدة بيانات علاقية. يمكن أن تجد مشغلات ODBC لهذه التطبيقات لكنها ذات تكلفة عالية. ما هو أسوأ من ذلك هو أن لديك الآن المزيد من طبقات البرامج. ولهذا السبب، يمكن الحصول على تحسينات كثيرة للخصائص والأداء بالتحول من ODBC إلى OLE-DB.

تحذير كن على حذر كبير عند تركيبك لتطبيقات تجارية تدعم ODBC. ففي حالة واحدة على الأقل صادفني قريبا، قام التطبيق بالكتابة فوق مدير مشغلات ODBC ومشغل أوراكل دون إعلام المستخدم. ولم يمكن استخدام مدير مشغلات ODBC كما توقف تطبيق بيسك موجود على الجهاز عن العمل. إذا قمت بتركيب تطبيقات تجارية تخص ODBC فيجب عليك أن تتأكد من اختبارها على جهاز تجريبي أولا!

في هذا القسم، سنبحث عن قرب في ODBC. ستشمل النقاط التي سنناقشها ما يلي:

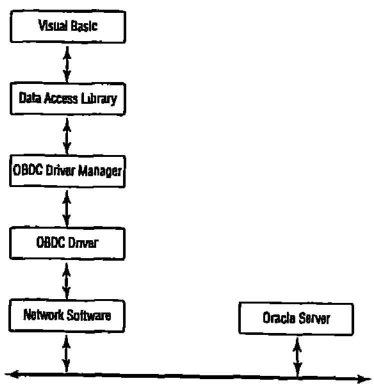
- أصول ODBC.
- مشغلات ODBC.
- فئات مشغلات ODBC.
- مستويات توافق ODBC.
- اختيار مشغل ODBC
- إعداد مصدر بيانات ODBC
- اختبار وصلة ODBC.

أصول ODBC

ODBC هي اختصار Open Database Connectivity أو إتصالية قاعدة البيانات المفتوحة، وقد تم تكوينها لبنية توفير طريقة لبرامج التطبيقات "مثل فيجوال بيسك" للاتصال بقواعد البيانات المختلفة للمزودين. وقد كونتها مجموعة تسمى مجموعة SQL أكسس التي لها ممثلون من معظم مزودي قواعد البيانات بما فيهم IBM و Sybase ومايكروسوفت وأوراكل وDigital Equipment Corporation. وبحلول عام ١٩٩٠، قدموا نموذجا منقحا يعتمد على SQL لواجهة تطبيق على مستوى الاستدعاء. وقد تم تطويره بشكل ملحوظ خلال الأعوام اللاحقة.

وفيما يتعلق بجهاز المزود فهناك العديد من طبقات البرامج التي تستخدمها دون علم بها، وهي موضحة في الشكل ١١-٥ بطريقة عرض شبكية مبسطة.

شكل ١١-٥



عــرض ODBC
لوصلة أوراكـــل
الخاصة بك.

مشغلات ODBC

أحد أكثر أسباب الارتباك في ODBC هو اكتشاف ما يناسب العمل مـــع مشـــغلك وقـــاعدة بيانـــاتك. وكقارئ لهذا الكتاب، فربما تكون قد لاحظت أن معظم ما هو مكتوب إما من وجهة نظر خـــادم SQL أو من نظرة عامة. فلنلق نظرة على ما يمكن أن تقدمه ODBC للمطور ثم نرى مـــا ينطبـــق علـــى أوراكل ولماذا.

فئات مشغلات ODBC

يتم استخدام ثلاثة فئات لتحديد مشغل ODBC. تعتمد هذه الفئات على البنية التي يدعمها المشغل.

طبقة واحدة مشغلات الطبقة الواحدة بنية لقواعد البيانات التي لا تستطيع كيفية أوامر SQL. هي أنظمة تعتمد أساسا على الملفات مثل مايكروسوفت أكســـس وFoxpro لكنـــها أيضـــا تتضمن إكسل. وقواعد البيانات هذه بها عجز، فليس لها أية فاعلية فـــي تشـــغيل الـــبرامج. ومهمة مشغلك هي معرفة كيف يستخرج البيانات من قاعدة البيانات وأيـــن ومتـــى تعمـــل خاصية الإقفال. يأخذ المشغل أمر SQL ويقوم بتحويله إلى سلسلة مـــن اســـتدعاءات API المناسبة لقاعدة البيانات.

طبقتين مشغلات ODBC ثنائية الطبقات تستعمل لبيئات المزود/الخادم الحقيقية، فهي مصممة لقواعد البيانات النشطة التي تتصل بواسطة SQL. يكون هناك غالبا بعض الحركة والتحليـل لأوامر SQL التي ترسلها لها أولا قبل أن تصل إلى قاعدة البيانات. يكون هذا ضروريا لأن قواعد البيانات غالبا ما يكون لها أشكال خاصة بها من SQL. ستحدث هذه الحركة عندمـــا تقوم بتمرير استعلام من فيجوال بيسك لأوراكل. إذا كنت تريد استخدام أي صيغة خاصـــة بأوراكل فعليك إذن تجاوز هذه الحركة وتحليل وتمرير جمل SQL مباشـــرة إلـــى قـــاعدة البيانات. تستخدم خيار SQLPassThrough لفعل ذلك، وهو الخيار الذي سنناقشه بالتفصيل في فصول لاحقة. يرسل المشغل ثنائي الطبقات النموذجي بيانات من التطبيق إلـــى قـــاعدة البيانات. سيقوم جهاز المزود بتشغيل التطبيق ومشغل ODBC ومدير المشغلات بينما يكـون الخادم على جهاز منفصل مع تشغيل خادم أوراكل. سيقوم الجهازان بتشغيل برامج شـــبكات

متوافقة.

ثلاث طبقات أقل أنواع المشغلات شيوعا هو مشغل ODBC ثلاثي الطبقات أو المشغل بشكل نموذجي عندما يكون هناك نوع من جيت واي بين خادم قاعدة البيانات والمزود.

ملاحظة فيما يتعلق بهذا الكتاب، فنحن مهتمون فقط في المشغلات ثنائية الطبقات. لاحظ أنه على الرغم من دلالة الاسم، فيمكن استخدامها بسهولة في بنية متعددة الطبقات لفيجوال بيسك. في هذه الحالة، هناك رابط ثنائي الطبقة بين قاعدة البيانات والمكونات متوسطة الطبقات.

مستويات توافق ODBC

بالإضافة إلى عدد الطبقات المصمم له مشغل ODBC، فإن هناك تصنيفا للمشغلات طبقا لمستوى توافقها. ومستوى التوافق هو مقياس لدرجة مواصفات ODBC API التي يدعمها المشغل. هناك ثلاثة مستويات يجب أن تعرفها:

مستوى النواة يجب أن يسمح لك المشغل بالاتصال بقاعدة بيانات وإعداد وتنفيذ جمل SQL واسترجاع مجموعات نتائج من استعلامات واستخدام جمل Commit و ROLLBACK للتحكم في الحركة واسترجاع الأخطاء.

المستوى الأول يجب أن يسمح المشغل بكل ما في مستوى النواة بالإضافة إلى توفير مربعات حوار خاصة بالمشغلات للاتصال بقاعدة البيانات وإمكانية الحصول على معلومات عن المشغل ومصدر البيانات.

المستوى الثاني يجب أن يسمح المشغل بكل ما في المستويين السابقين بالإضافة إلى السماح لك بسرد واستعراض إمكانيات مصادر البيانات، واسترجاع مجموعات نتائج في صيغ مختلفة، وتوفير خلايا قابلة للإزاحة. يجب أيضا أن يسمح لك باسترجاع معلومات دلائل إضافية من قاعدة البيانات.

إذا تم تحديد مشغل ODBC بمستوى معين، فيجب أن يوفر جميع خصائص هذا المستوى. هذا لا

يعني أن مشغل المستوى الأول على سبيل المثال لا يستطيع توفير بعض خصائص المستوى الثـاني. في الحقيقة، توفر معظم المشغلات الخلية القابلة للإزاحة. وبسبب هذا، فقد تـم اسـتخدام مشـغلات المستوى الأول في بعض التطوير المعقد ولذلك لا يجب أن يتم استبعادها.

وعلى الرغم من أن مستويات التوافق متخصصة، فلا تتم ترجمتها دائما تطبيق. هناك الكثير مـن المشغلات التي تم تعريفها على أنها مشغلات المستوى الثاني لكنها لم تعمل بنفس المستوى. لم تكـن هذه مشكلة حتى تم تقديم بعض الأدوات مثل وحدات البيانات البعيدة (RDO) للمطورين، وهي أدوات تتطلب مشغل من المستوى الثاني لخصائصها الأكثر تطورا. إذا اخترت المشغل الخطأ، فمن المحتمل أن ترى الرسالة المفرغة "Driver not Capable".

تلميح

يجب أن تحاول الحصول علـى مشـغلات ODBC حديثـة نسـبيا. هـذه المشغلات هي الأكثر احتمالا ليكون لها مستويات التوافق العالي وتمكنـك من استخدام معظم خصائص فيجوال بيسك. لكن تأكد من اختبارهـا علـى جهاز تجريبي قبل أن تبدأ في استعمالها.

اختيار مشغل ODBC

والآن بعد أن عرفنا أنواع مشغلات ODBC الموجودة وإمكانياتها، فقد حان وقت اختيار أحدهما للعمل مع قاعدة بيانات أوراكل الخاصة بك. أحد الجوانب المحيطة لـ ODBC هو إيجاد مشغل ينفـذ مـا تريد. يوفر الكثير من المشغلات سهولة كبيرة للتعامل مع مستويات التوافق المتنوعة، لكن كل مشـغل جديد تكون له إمكانيات مختلفة قليلا أو ربما تؤثر على إمكانيات المشغلات التي قمت بتركيبها أخـيرا على جهازك.ومع سرعة تغير الأشياء من حولنا، يكون من الصعـب أن نحـدد المشـغلات التـي نستخدمها والمشغلات التي نتجنبها. بالتأكيد يكون من الأسهل التمسك بالمشغلات التـي تـأتي مـع أوراكل (بواسطة Intersolv) أو من مايكروسوفت وفهم خصائصها والتعايش معها.

نصيحتي هي استخدام واحد من مشغلات ODBC لهاتين الشركتين لكن عليك أن تحذر قبل التجديد لا تحصل على نسخ من جميع الإصدارات الأحدث ثم تقوم بتركيبها دون تميز ودون تصور أنها قـد تتداخل مع بعضها البعض. إذا كان لديك جهاز تجريبي، قم بتركيب أي مشغل جديد عليه وتأكد مـن عمله كما يتم تعريفه وأنه لا يتداخل مع مشغلات أخرى. أحيانا تكون الطريقة الوحيدة للخروج مـن موقف كهذا هي إعارة تهيئة القرص والبداية من جديد.

هناك حالة واحدة بالتحديد هي المشغل الذي جاء كجزء من MDAC ١,٥، مكونات الوصول للبيانات من مايكروسوفت، التي يوفر خصائص لمكونات البيانات النشطة (ADO). كشف المشغل الجديد عن خطأ في RDO والذي لم يتم إصلاحه حتى فيجوال بيسك ٦.

ملاحظة يبدو أن مشغل مايكروسوفت ODBC قد أخذ وقتا وجهدا أكثر ليتم تطويره.

مصادر مشغلات ODBC

العديد من مصادر مشغلات ODBC المحتملة متاحة لأوراكل. على أي حال، إذا كنت تريد شراء مشغل أقل استخداما، مستحضرا اختياراتك في أثنين: واحد من مايكروسوفت وواحد من أوراكل.

مشغل ODBC من مايكروسوفت

يتاح مشغل ODBC من مايكروسوفت مع فيجوال بيسك على الرغم من أنك قد تجد أنه قد أصبح قديما بسرعة. تفضل الإصدارة التي مع فيجوال بيسك ٦ عن أي إصدارات سابقة.

مشغل ODBC من أوراكل

تم تطوير مشغل ODBC من أوراكل بواسطة Intersolv. وهو يوجد في مجموعات من منتجات أوراكل المتعددة بما فيها الخوادم. أيضا من المحتمل أن تصبح هذه المشغلات قديمة بسرعة، ومن الفضل تحميل النسخ الأحدث من الويب فقط كن مستعدا لاختبار المشغل تماما قبل استخدامه. العنوان على الويب هو www.oracle.com/products/free_software/index.html.

يجب عليك استخدام مشغلات أوراكل٨ إذا كنت تخطط التركيب Net8 والإصدارة ٨ أو احدث من واجهة تطبيق استدعاء أوراكل. استخدم مشغلات أوراكل ٧ و SQL*Net لغير ذلك.

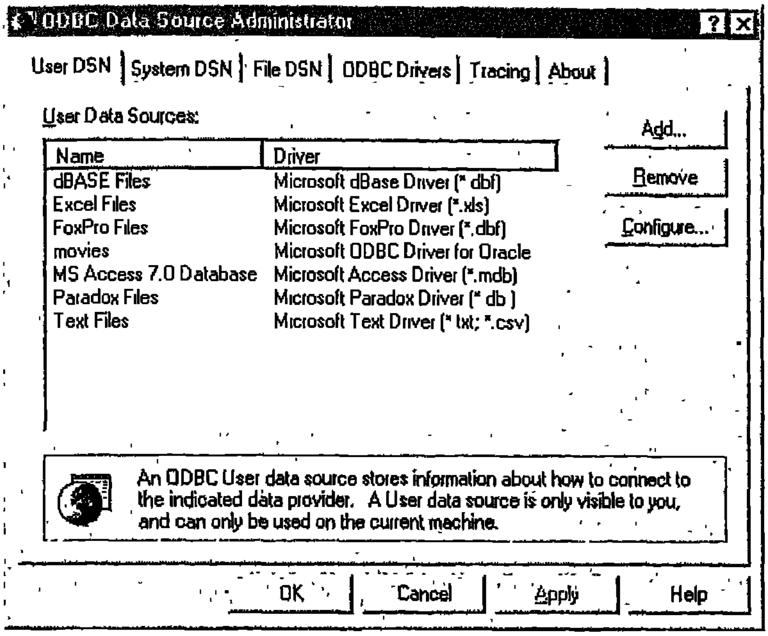
إعداد مصدر بيانات ODBC

بعد اختيار مشغلك فقد حان وقت تركيبه وإعداد مصدر بيانات. لتفعل ذلك، تستخدم برنامج مسئول مصادر بيانات ODBC (ODBCAD32.EXE) الذي يجب أن يكون موجودا على جهازك في Control Panel، "ODBC ٣٢ بيت". لتشغيل هذا على ويندوز ٩٥ أو NT، اتبع الخطوات التالية:

١- انقر زر Start واختر Setting ⇦ Control Panel.

٢- انقر نقرة مزدوجة على "ODBC".

سترى نافذة مثل الموضحة في الشكل ١١-٦. ستلاحظ عدة علامات جدولة وهي العلامات المفصلة في الجدول ١١-١.



شكل ١١-٦ نافذة دخول لمشغل مايكروسوفت.

الجدول ١١-١: خيارات مسئول مصادر بيانات ODBC.

رأس العلامة	بيانها
User DSN	يسرد العنوان أسماء مصادر البيانات المعدة للمستخدم الحالي.
System DSN	يسرد العنوان أسماء مصادر البيانات المعدة للنظام بدلا من مستخدم

الجدول ١١-١: خيارات مسئول مصادر بيانات ODBC.

رأس العلامة	بيانها
	مفرد.
File DSN	يسرد العنوان مصادر البيانات المعتمدة على ملفات.
ODBC Drivers	يسرد العنوان مشغلات ODBC المركبة على الجهاز المحلي.
Tracing	يتحكم هذا العنوان في تتبع مدير مشغلات ODBC لاستدعاءات SQL.
About	يسرد هذا العنوان اسم الإصدارة والملف لجميع المكتبات الديناميكية الرئيسية لـ ODBC. يجب أن تحتفظ بهذه المعلومة في وثائقك لأن المشغلات يمكن الكتابة فوقها بواسطة برامج أخرى.

ملاحظة إذا كنت تشغل تطبيق مزود ويندوز NT الذي يصل إلى مشغل ODBC ويجب تشغيل التطبيق كخدمة، إذن يجب عليك إعداد اسم مصدر بيانات للنظام System DSN. سيمكن تشغيله حتى إذا لم يدخل أي مستخدم.

لإضافة اسم مصدر بيانات جديد لأوراكل. يجب أن نتأكد أولا من تركيب مشغل أوراكل. يمكنك رؤية المشغلات التي تم تركيبها من علامة جدولة ODBC Drivers. إذا لم يكن مركبا، سيكون عليك الحصول على برنامج المشغل من مزودك الخاص وتركيبه. بعد تركيبك للمشغل، يجب أن تختار علامة جدولة اسم مصدر بيانات المستخدم User DSN والنقر على زر Add. يسمح لك هذا الزر بإضافة اسم مصدر بيانات جديد. عندما تنقر زر Add سترى قائمة بكل المشغلات المتاحة. اختر المشغل الذي تريده وانقر نقره مزدوجة عليه. والآن سيكون عليك ملء بيان باسم مصدر البيانات الذي تريد إعداده (راجع الشكل ١١-٧). هناك عادة ثلاثة أو أربعة حقول يجب عليك ملئها.

اسم مصدر البيانات: يمكنك أن تعطي أي اسم لمصدر البيانات على الرغم من أفضلية أن تجعله واضح المعنى. تستطيع إعداد أي رقم من أسماء مصادر البيانات لنفس مشغل ODBC. عند

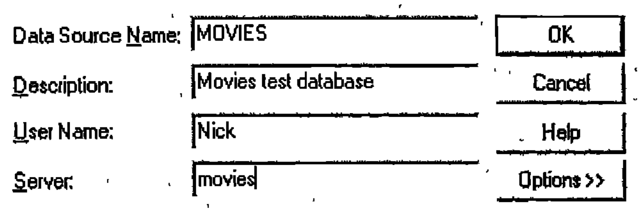
التطبيق، يكون من الأبسط استخدام نفس اسم خدمة قاعدة بيانات أوراكل.

بيان استخدام أي بيان تريده هنا: لن يرى المستخدمون هذا الحقل أبدا، لذلك تستطيع تركه خاليا إذا أردت بالرغم من انه قد يفيدك إذا كنت قد أعددت أسماء مصادر بيانات كثيرة.

اسم المستخدم: هذا هو اسم المستخدم الذي تريده أن يظهر في مربع حور الدخول عند بداية التشغيل يستطيع المستخدمون تغليب هذا الحقل عند دخولهم.

خادم: هذا هو اسم خدمة أوراكل أو اسم قاعد البيانات البديل الذي رأيناه في قسم "إعداد Net8 و SQL*Net" ويجب أن يتطابق مع الاسم الذي أعددته في ملف Tnsames.ora. تذكر أن الوظيفة الأساسية من هذا الإعداد هي توفير معلومات لتحويل اسم مصدر البيانات إلى اسم خدمة قاعدة بيانات. يمكن أن تتماثل جميع هذه الأسماء إذا أردت "وهذا أسهل شيء" لكنها لا يجب أن تكون مماثلة. لاحظ أن هذا الحقل يمكن أن يسمى اسما آخر "مثل Connect" طبقا لوظيفة وإصدارة المشغل.

شكل ١١-٧
إعداد اسم مصدر بيانات ODBC



ملاحظة لتركيب مشغل ODBC أوراكل، تحتاج إلى وجود جميع برامج الشبكات المطلوبة بالفعل. بدون هذه البرامج، ستظهر لك رسالة خطأ.

تكوين سلسلة وصلة ODBC

سلسلة الوصلات هي سلسلة تتضمن معلومات كافية لتطبيق فيجوال بيسك الخاص بك لعمل وصلة بقاعدة بيانات أوراكل. يتم تمرير السلسلة إلى مدير مشغلات ODBC الذي يتولى المسئولية عندئذ لعمل الوصلة. سيكون عليك استخدام سلسلة الوصلة هذه بشكل طبيعي عندما تتصل بقاعدة البيانات لأول مرة بواسطة Open Database أو طريقة مشابهة. ستستخدمها أيضا لعناصر التحكم في

البيانات في بيئة أكسس.

هناك أجزاء متعددة اختيارية لسلسلة الوصل ويمكن أن تختلف البيانات المحتملة لمشغلات ODBC المختلفة تتكون السلسلة نفسها من ازدواج من كلمات أساسية وقيم يوجد بينهما فواصل. الكلمات الأساسية المحتملة موضحة في الجدول ١١-٢.

الجدول ١١-٢: الكلمات الأساسية في سلسلة وصل ODBC.

الكلمات الأساسية	بياناتها
ODBC	تحدد السلسلة كسلسلة وصل ODBC، ولا توجد لهذه الكلمة الأساسية قيمة ملحقة.
DATABASE	اسم قاعدة البيانات. تفيد هذه الكلمة الأساسية أصلا في تحديد الملف mdb. في قاعدة بيانات أكسس.
UID	اسم المستخدم الافتراضي لمصدر البيانات.
PWD	كلمة السر لاسم المستخدم الافتراضي. تضمين هذه القيمة هي مسألة تأمين ويجب تجنبها.
DSN	اسم مصدر البيانات الذي أعددته في تطبيق مسئول ODBC. يتم تحويل هذا الاسم في أوراكل إلى بديل قاعدة بيانات.
LOGINTIMEOUT	الوقت بالثواني لمحاولة الدخول قبل انتهاء الوقت. الافتراضي هو ٢٠ ثانية. لاحظ أن هذه المعلومة يمكن تخزينها أيضا في التسجيل Registry.

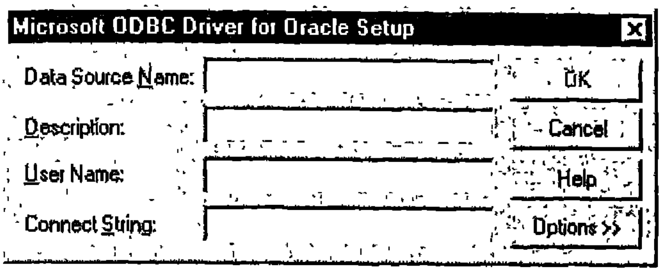
ملاحظة عندما تستخدم OLE-DB، سيكون لديك معاملات أخرى متاحة لك. تتم مناقشة هذا الموضوع في فصل ١٧، وحدات بيانات Activex.

نظريا، أنت لا تحتاج لتزويد هذه الكلمات الأساسية حيث سينبهك مدير مشغلات ODBC عن القيم المفقودة عند التشغيل. إما عمليا على أي حال، فلن يفيد هذا المستخدمين النهائيين. يجب أن تحدد اسم

مصدر البيانات حتى يتأكد المستخدمون من قاعدة البيانات التي يتصلون بها. سيكون مثال لسلسلة الوصل الخاصة بك كهذا:

strConnect = "ODBC;DSN=Movies"

عندما يتم تشغيل التطبيق بسلسلة الوصل هذه، ستتم المطالبة باسم المستخدم وكلمة السر في مربع حوار دخول كالموضح في الشكل ١١-٨. لاحظ أن كل مشغل ODBC يمكن أن يكون له نافذة الدخول الخاصة به، لذلك قد تجد بعض الفروق.



الشكل ١١-٨: نافذة دخول لمشغل ODBC مايكروسوفت.

إذا كنت تقوم بتطوير تطبيقك، فربما لن تريد الاستمرار في إدخال نفس المعلومات في مربع الدخول، لذلك تستطيع عمل ترميز دائم لاسم المستخدم الخاص بك وربما لكلمة السر في سلسلة الوصل. يجب أن تعلم أن الترميز دائم لاسم المستخدم الخاص بك وربما لكلمة السر في سلسلة الوصل. يجب أن تعلم أن الترميز الدائم لهذه المعلومات اختياريا كإجراء أمني. مثال لسلسلة بهذه الكلمات الأساسية هو:

strConnect = "ODBC;DSN=Movies;UID=nick;PWD=ummagumma"

بافتراض صلاحية اسم المستخدم وكلمة السر، لن يكون عليك أن ترى نافذة الدخول على الإطلاق.

مربع حوار الدخول الخاص بك

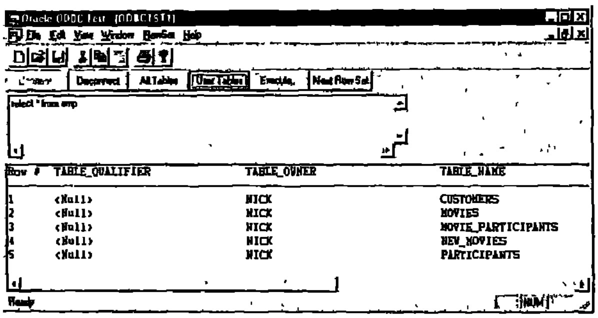
تقدم مشغلات ODBC كثيرة مربع حوار يطالب بأكثر من مجرد اسم المستخدم وكلمة السر تطالبك بعض المشغلات باسم مصدر البيانات أيضا. يمكن لهذه المعلومات أن تربك المستخدمين تماما وربما دفعتهم للدخول إلى قاعدة

البيانات الخطأ. أحد إصدارات مشغلات ODBC قمت أنا باستخدامه كان يطلب اسم البيانات كل مرة يتم عمل وصلة. ولأن برامج فيجوال بيسك يمكن أن تستخدم أكثر من وصلة، فإن ذلك قد أدى إلى إجراء تطبيق فيجوال بيسك لوصلتين مع اتصال كل منهما بقاعدة بيانات مختلفة بغير قصد.

عند هذه النقطة، بدأت أتجاوز مربع حوار الدخول الافتراضي نهائيا. يتم تعرضه فقط عند فقد أي معلومات، ومن السهل إنشاء مربع حوار VB يطلب اسم المستخدم وكلمة السر ثم يقوم بالدخول. إذا كنت تستخدم بنية متعددة الطبقات، فستحتاج استخدام هذا الأسلوب على أي حال، وإلا سيظهر مربع الدخول على خادم مكون.

اختبار وصلة ODBC

بمجرد إعدادك Net8 أو SQL*Net واختبارك لهما تماما ثم تركيبك وتكوينك لمشغل ODBC الذي اخترته، يجب أن تكون خطوتك التالية هي المشغل بفتح وصلة مع قاعدة البيانات. تستطيع فعل ذلك بواسطة تطبيق فيجوال بيسك، لكن الطريقة الأبسط هي استخدام تطبيق صغير مصمم لهذا الغرض يسمى ODBC T32 EXE وهو الموضح في الشكل ١١-٩.



الشكل ١١-٩: برنامج اختبار ODBC.

لاستخدام برنامج اختبار ODBC، يجب أولا أن تتصل بقاعدة بيانات أوراكل. ولتقوم بهذه الخطوة، انقر زر Connect واختر واحدا من مصادر بيانات SQL. سيكون مصدر البيانات هو نفسه الذي قمت بإعداده في القسم السابق، وبمجرد اختيارك لاسم مصدر البيانات، سترى مربع حوار الدخول المعروف الذي رأيناه في الشكل ١١-٨.

يمكنك إجراء اختبار بسيط لتكوينك بعرض الجداول الصالحة لوصولك لها. تستطيع القيام بهذه الخطوة بنقر زر User Tables وهو ما يؤدي بك إلى المعرض الذي رأيناه في الشكل ١١-٩. الجداول المعروضة هي الجدول التي لديك في مخطط قاعدة بياناتك. سترى فقط خمسة صفوف معروضة افتراضيا وهي كل ما تحتاجه لمراجعة الوصلة، لكنك تستطيع استخدام خيار القائمة Row Set لتغيير هذا العدد. تستطيع أيضا تنفيذ أوامر SQL على قاعدة البيانات. على أي حال، ليست هذه الأداة بديلا لـ SQL*Plus فهي تحتوي على نطاق محدود من الأوامر التي يمكنك تنفيذها، وكذلك فهي ليست أداة إدارية نموذجية.*

مشاكل محتملة مع إعداد ODBC

كما ذكرنا سابقا في هذا الفصل، يجب أن تعمل طبقا لبعض الخطوات لإعداد واختبار وصلتك بقاعدة البيانات. يجب أن تأتي اختبارك لإعداد ODBC بعد اختبار الخادم وتركيب شبكة المزود. ومن الغريب حقا أن الكثيرين يريدون تجاوز هذه الخطوات على الرغم من الطريقة الأصلية ستكون هي الأفضل على المدى الطويل.

الشيء التالي الواجب اختباره هو ما إذا كنت تشغل برامج متوافقة. ويكون الاهتمام الأساسي بمعرفة نوع البرامج هل هي ٣٢ بيت كلهما. لسوء الحظ، لا يستطيع مشغل ٣٢ بيت التعامل مع نسخة ١٦ بيت من SQL*Net.

إذا كانت كل خطوة في السلسلة تعمل حتى هذه النقطة، فستكون المشكلة الرئيسية التي تواجهها هي إذا ما كان تركيب ODBC قد تم بشكل صحيح أم لا، وإذا كنت قد أعددت اسم مصدر بيانات ODBC لاسم خدمة أوراكل الخطأ. يجب مراجعة هاتين النقطتين بدقة. إذا استمرت المشكلة موجودة، يجب أن ترجع أولا لمراجعة برامج الشبكة مرة أخرى وإذا لم ينجح هذا، يجب عليك إعادة تركيب برنامج الـ ODBC.

تحذير هناك مشكلة صادفتني قريبا كانت مع برنامج أوراكل. تستطيع الإصدارات السابقة الكتابة فوق ملفات OCX وDLL. ويقوم برنامج التركيب بفعل هذا دون إخطارك ويستطيع الكتابة فوق الملفات ذات الإصدارات الأقدم. والمشكلة التي لا حل لها هي أنك لا تستطيع إنشاء وصلات مرة أخرى بعد تركيب بعض البرامج الجديدة. ويكون الحل الوحيد أمامك هو إلغاء تركيب البرامج والبداية من جديد.

OLE-DB

أحدث بدائل ODBC من مايكروسوفت قد جاء لسببين كبيرين: الأول هـو الحاجـة لتقنيـة وصـول للبيانات تعتمد على نموذج وحدات المكونات وهي ما ستجعل التقنية متاحة لتصبح عبر الحاسبات بدلا من أن تكون محصورة ببيئة ويندوز. ولجعل طريقة وصولك للبيانات مستخدمة عبر حاسبات مختلفة فإن ذلك يتطلب أن يكون نموذج وحدات المكونات (COM) متاحا بسهولة عبر الحواسب. الإصـدارة الموزعة من Com وهي Dcom في منافسة Common Object Request Broker CORBA) (Architecture من مجموعة المزودين بخلاف مايكروسوفت.

السبب الثاني للابتعاد عن ODBC هو أنه على الرغم من تصميـم ODBC كطريقـة للوصـول للبيانات لقواعد البيانات العلاقية (مثل أوراكل وخادم SQL)، فإن هناك حاجة للقدرة على الوصـول لمصادر بيانات أخرى غير المصادر العلاقية. وكل مصدر، سواء كان نظام بريد إلكتروني أو قـاعدة بيانات للوحدات أو ملفات HTML أو نظام تدفق عمل أوفيس، يحتاج إلى طـرق وصولـه الخاصـة وطرق جديدة لربط مصادر البيانات المتنوعة سويا. وبنية ODBC غـير فعالـة للوصـول الكـافي لمصادر البيانات المتنوعة، وقد أعدت مايكروسوفت خططا لتغيرها وإبدالها بالتقنية العالمية للوصـول للبيانات (UDA) كجزء من خطط بنية الشبكة الموزعة للسنوات القليلة القادمة.

هناك طريقتان مختلفتان تماما للوصول للأنواع المختلفة من مصادر البيانات. الطريقة الأولى هـي إحضار جميع البيانات إلى قاعدة البيانات وهو ما يتطلب أن تكون قاعدة البيانات قابلة للتمريـر مـع أدوات متنوعة كثيرة للوصول إلى البيانات ومعالجتها. وهذه هي طريقة أوراكل الطبيعية، وتسـتطيع قاعدة البيانات التعامل مع البيانات الصوتية والحركية والرسومية والمساحية بالفعل بالإضافـة إلـى الإمكانيات المتقدمة للتعامل مع النصوص (يسمح خيار Context بالأبحـاث الأساسـية والتقاربيـة بالإضافة إلى الفهارس الموزعة الكاملة). يتناسب هذه الطريقة العالمية تماما مع بيئة مستودع البيانات التي يتم فيها إرسال بياناتك مؤسسة عملك تماما إلى قاعدة البيانات دوريا.

الطريقة البديلة هي بناء مجموعة أدوات تزود بنفس الإمكانيات لكنها تمتد إلى مصـادر البيانـات الفردية تحتاج هذه الطريقة لن تتكون من طبقات لها مكونات تعمل بشكل متفاعل وتوفر واجهة تطبيق لمصادر البيانات بالإضافة إلى السماح بمعالجة البيانات والاستعلام عنها بمجـرد تقديمـها بطريقـة موحدة. أخذت مايكروسوفت بهذه الطريقة. وعلى الرغم من أن هذين أسلوبان متنافسان فإن هـذا لا ينفي إمكانية عملهما معا. في الوقع، فإن طريقة المكون يمكنها تكميل طريقة قاعدة البيانات وتوفير أية إمكانيات غائبة تعتقد أنك تحتاجها. وكمستخدم أوراكل، تستطيع الحصـول علـى أفضـل مـا فـي

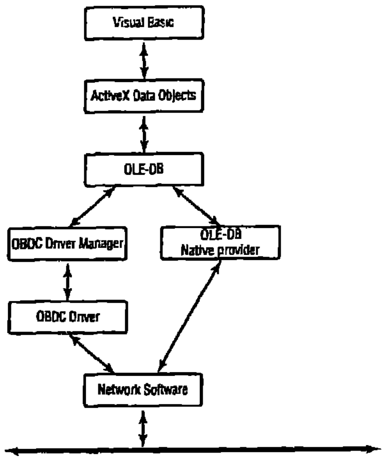
الطريقتين.

تم تصميم UDA للأداء المتطور والقابلية للتحجيم بالإضافة إلى المرونة والاستقرار المتطور لنظام أبسط مع أجزاء أقل قابلة للتعطيل. الجزء الهام في UDA بالنسبة لنا كمطوري أوراكل وفيجوال بيسك هو أن طبقة ODBC إبدالها بـ OLD-DB وهي بنية تتكون من واجهات تطبيق للوصول للبيانات على مستوى النظام. وOLD-DB ليست مجهزة للوصول من فيجوال بيسك لكنها ليس مطلوبا منها ذلك. وعلى الرغم من أنها كانت متوفرة مع MDAC ١,٥ فإن فيجوال بيسك ٦ يزودنا بالنماذج الجديدة لوحدات بيانات Activex (ADO) وخدمات البيانات البعيدة (RDS). سنناقش هذه النماذج بالتفصيل في فصل ١٨. يوفر فيجوال بيسك ٦ أيضا مزود OLE-DB أساسي.

OLE-DB هي مجموعة من واجهات تطبيق تعتمد على المكونات لإدارة البيانات (راجع الشكل ١١-١٠). وهي تعتمد على COM وتتكون من ثلاث فئات من واجهات التطبيق:

- **مزودات البيانات** وهي وجهات لمصدر البيانات الحقيقي. على سبيل المثال، ستعرض الصفوف والأعمدة الموجودة في جدول أوراكل. ومهما كان مصدر البيانات الذي تستخدمه، فسيتم عرض البيانات كمجموعات صفوف للمكونات التي تستخدمها. تؤدي مزودات البيانات وظيفة كل مدير مشغلات ODBC والمشغلات الخاصة بقواعد البيانات.
- **مستخدمي البيانات** أي مكون يصل للبيانات من مزود بيانات يعتبر مستخدم بيانات. في هذه الحالة، تؤدي وحدات بيانات Activex هذه الوظيفة.
- **مكونات الخدمات** وهي مكونات تقوم بتوفير نوع من إمكانية المعالجة للنظام مثل معالج الاستعلام أو مدير الخلايا أو نظام إدارة الحركة. لن يكون معالج الاستعلام مطلوبا إذا كنت تحاول الوصول للبيانات فقط من قاعدة بيانات أوراكل لأن معالج الاستعلام الأساسي سيكون كافيا تماما. على أي حال، عند استخدام مصادر بيانات بدون معالجات استعلامها الخاصة أو عند ربط بيانات أوراكل ومصدر بيانات آخر، تصبح هذه الإمكانية أكثر أهمية.

أوليا، كان مزود البيانات الوحيد المتاح لـ OLD-DB هو مزود ODBC. وقد أدى ذلك للموقف غير العادي الذي تم فيه إنشاء واجهة تطبيق إضافية على ODBC مع فوائد ضئيلة. في الواقع، يمكن رؤية بعض التغييرات في الأداء بالفعل وهي ليست كلها سيئة. في الوقت المناسب، سيقدم عدد كبير من مزودات البيانات الأساسية أداء وفاعلية متطورة. ستشهد مزودات البيانات تغييرات كثيرة خلال السنوات القليلة القادمة.



شكل ١١-١٠
مزودات بيانات لـ ODBC OLE-DB سيتم إبدالها بمزودات أساسية.

خلايا OLE-DB

في القسم التالي، "الخلايا"، سنرى عرضا عاما لتنفيذ الخلايا. من عدة جهات، فإن الساليب في OLE-DB كما هي في ODBC إلا أنها موجودة الآن في مكونات الخدمات وليس في مشغلات ODBC. تضاف مكونات الخدمات هذه لمزودات البيانات لتحسين الفاعلية. على سبيل المثال، لأوراكل خلية مجموعة صفوف للأمام فقط وهي التي حلت محل الخلية Firehose. وكما رأينا مع خلايا ODBC، يتوقع الكثير من المستخدمين المزيد من الإمكانيات التي توفرها (مدير الخلايا) نفس الأساليب كما في ODBC لوضع البيانات على الذاكرة الوسيطة للمزود.

لا يقوم مدير الخلايا هذا بعملية السحب والتحديث للبيانات، فهذه الوظائف يؤديها مركب خدمات آخر يسمى خدمة التزامن. تعني بنية المركب أن المركبات المختلفة تستطيع العمل على طبقات مختلفة. وبسبب هذه الغمكانية، فمن الممكن الآن فك ارتباط الخلية من إجراء التزامن واستخدام مجموعات السجلات غير الموصولة التي تأتي مع خدمات البيانات البعيدة.

ملاحظة

مزود OLD-DB الأساسي لأوراكل المتوفر مع فيجوال بيسك ٦ يدعم فقط خلية للأمام فقط.

الخلايا

قد تكون المفاهيم الخاصة بالخلايا غامضة لمن لا يعلمها. هناك مشكلة هي أن كل مزود قواعد بيانات

يكون لديه RDBMS بإمكانيات مختلفة وطرق مختلفة لتنفيذ المهام. ومشكلة أخرى أننا نصادف الخلايا على جانب الخادم بالإضافة إلى الخلايا على المزود، الخلايا على المزود، وهي أشياء منفصلة تماما.

وسواء كنت تتحدث عن خلايا الخادم أو الخلايا على المزود، فإن لهما خصائص مشتركة. نعرف جميعا مفهوم الخلايا التي تشير إلى مكاننا الحالي في وثيقة نصية. وفي مصطلحات قاعدة البيانات، لا تختلف الخلايا كثيرا فهي مؤشرات على مكاننا الحالي في مجموعة نتائج وهي مجموعة من الصفوف والأعمدة المسترجعة من قاعدة البيانات.

عندما تسترجع بيانات من أوراكل بواسطة جملة SELECT، تحدد قاعدة البيانات أولا جزءا من مساحة SQL المشتركة لـ SQL المراد تحليلها ويتم إنشاء خطة تنفيذ. أية بيانات ليست في ذاكرة خادم أوراكل يتم إدخالها ويتم إنشاء مجموعة النتائج وإرجاعها للمستخدم صفا صفا. إذا تجاهلنا مفاهيم متغيرات الخلايا، فليس هناك خلية جانب الخادم كتلك. يتم إرجاع الصفوف واحدا واحدا دون رجوع، وعند استرجاع القيمة الأخيرة، تتم استعادة مساحة الذاكرة التي تشمل مجموعة النتائج إلى النظام. إذا أردت قيمة سابقة، فسيكون عليك إعادة تنفيذ جملة SELECT. يصبح هذا واضحا عند استخدامك للخلايا في PL/SQL.

ملاحظة ملاحظة سريعة على المصطلحات. مجموعة النتائج هو المصطلح الذي يطلق على مجموعة الصفوف التي يتم إرجاعها من أوراكل لفيجوال بيسك. وهي تحتوي على واحدة أو أكثر من مجموعات الصفوف التي تمثل كل واحدة منها صفا. عندما تصل البيانات لفيجوال بيسك، تسمى مجموعة سجلات Recordset وحدات الوصول للبيانات وكمجموعة سجلات rdoResultset rdo في وحدات البيانات البعيدة.

خلية خادم أوراكل أحادية الاتجاه مفهوم هام لأنها تختلف عما اعتاد عليه معظم المطورين وما يتوقعونه من فيجوال بيسك. اعتدنا على رؤية مجموعة البيانات والتمرير للأمام والخلف خلالها وتغيير أي قيمة نريدها ببساطة.يجب أن تلاحظ هنا أن أوراكل ليس مختلفا في هذا الخصوص عن معظم التطبيقات الأخرى لـ RDBMS. فمن أين يأتي الاختلاف؟ يوفر مدير الخلايا هذه الفاعلية الإضافية، والتمرير هي خاصية تعمل على المزود لا على الخادم، على الرغم من أن هناك حدا لما يمكن توقعه. يوفر مدير الخلايا أربعة أنواع من الخلايا الموجودة أصلا في مواصفات ODBC وهي الخلايا الثابتة والديناميكية وخلايا المجموعات الأساسية والخلايا المختلطة، مع الإضافة الشائعة لخلية

Firehose. يتم تنفيذ نفس هذا النموذج في أساليب الوصول الأخرى (وحدات بيانات Activex ووحدات أوراكل لـ OLE)، على الرغم من أنه ليس كل مديري الخلايا توفر جميع الأنواع.

ملاحظة استخدم هنا كلمة مدير الخلايا بشكل عام. في ODBC يتم تنفيذ هذه الوظيفة بواسطة مشغل ODBC أو مدير المشغلات. وفي OLE DB يقوم بهذه الوظيفة مركب خدمة يسمى مدير الخلايا، ومع وحدات أوراكل لـ OLE، يتم توفير إمكانية الخلايا كجزء من الخادم العامل. وتقوم كلها بوضع نتائج الاستعلامات في صيغة مجموعة سجلات.

الخلايا الثابتة

الخلية الثابتة هي أبسط صورة لخلية. تكون صفوف مجموعة النتائج المسترجعة بما فيها أي كلمات أساسية وجميع القيم تكون ثابتة عند إنشاء خلية جانب المزود. تقوم مكتبة الخلايا التي تنشئ الخلية بقراءة جميع القيم بشكل صحيح وتضعهم في الذاكرة على جهاز المزود حتى نستطيع التمرير إلى الخلف والأمام بنفس سهولة التحرك لأعلى وأسفل في صفوف جدول بيانات. المشكلة في الخلايا الثابتة هي أنها ثابتة وهذا يعني أنه أثناء التمرير حول الخلية، تغفل مجموعة السجلات أية تغييرات تحدث في قاعدة البيانات. يمكن أن يتم حذف صفوف أو تغيير قيم بواسطة مستخدمين آخرين، لكن لن يظهر هذا على البيانات. لا تستطيع حتى تغيير القيم في مجموعة السجلات الخاصة بك، ولذلك فهي تنتج فقط خلية للقراءة فقط. يكون توافق الخلايا الثابتة عاليا جدا (تكون مجموعة النتائج مماثلة تماما)، ويكون التزامن عاليا بسبب التزامن الفائق لأوراكل. على أي حال، ربما تجد أن البيانات تصبح قديمة بسرعة.

ملاحظة خلية نوع اللقطة في وحدات الوصول للبيانات هي في الأساس خلية ثابتة.

الخلايا الديناميكية

على النقيض من الخلايا الثابتة توجد الخلايا الديناميكية. تشير هذه الخلايا لمجموعة من البيانات التي يتم تحديثها باستمرار وهو ما يعني أن الخلية تمثل بدقة أي تغييرات تقوم بإجرائها على البيانات

بالإضافة إلى جميع التغييرات التي قام بها جميع المستخدمين الآخرين. من الصعب تطبيق الخلايا الديناميكية وهي تعتمد بشكل عام على وجود خلايا جانب المزود، مثل تلك التي يوفرها خادم مايكروسوفت SQL. كما يمكنك أن تتخيل، فإن توافق البيانات ضعيف فلا تستطيع الاعتماد على شيء يظل كما هو لفترة طويلة على الرغم من أن البيانات حديثة قدر الإمكان. والخلايا الديناميكية هي أكثر الخلايا تكلفة للاستخدام من وجهة نظر تتعلق بموارد الخادم والمزود. إذا كنت لا تزال تذكر فصل ٢، بنية قاعدة بيانات أوراكل، سترى أن أوراكل يقدم عرضا متوافقا جدا للبيانات وهذا مالا يدفعها لإنتاج خلايا ديناميكية. وكنتيجة لذلك، لا تتاح هذه الخلايا لمطوري فيجوال بيسك الذين يعملون بأوراكل.

خلايا المجموعات الأساسية

خلايا المجموعات الأساسية هي وسط بين الخلايا. فيكون لها تمثيل ثابت من الصفوف في مجموعة نتائج (تخزن مع قيمة مفتاح فريدة في المجموعة الأساسية)، لكن قيم الصفوف يتم سحبها فقط قبل الاحتياج إليها مباشرة. والمجموعة الأساسية هي مجموعة من المؤشرات للصفوف الحقيقية في جدول أوراكل. وهي بهذا تمثل مطلوب مصدر هام على كل من الخادم والمزود. وبالرغم من أن هناك العديد من التطبيقات المختلفة لخلايا المجموعات الأساسية، فإنها تسمح لك بشكل عام برؤية أحدث القيم للصفوف بما فيها التحديثات المثبتة بواسطة مستخدمين آخرين. خلية المجموعة الأساسية هي الطريقة الوحيدة للحصول على خلية قابلة للتحديث مع أوراكل. على أي حال، يجب تنشيط فاعليتها من مكتبة الخلايا وهو ما يجعلها غير فعالة بشكل واضح. ليس لديك خيار وضعها على الخادم.

ملاحظة	خلية نوع المجموعة الديناميكية في وحدات الوصول للبيانات تشبه في مضمونها خلية المجموعة الأساسية.
تحذير	عند تحديث البيانات أو حذفها بواسطة مستخدم آخر، يتم الإشارة إليها بأنها داكنة "dirty". لا يسمح مشغل أوراكل لك برؤية التغيير، ولذلك فإذا كان مهما لتطبيقك ستريد التأكيد على أن تكتشف تعليمات أي تغييرات لصف هي على وشك تبديله. يقلل اكتشاف التغييرات مميزات مجموعات السجلات القابلة للتحديث ويجب على المستخدمين أن يقبلوا رسائل الخطأ التي يمكن أن تنتج من موقف كهذا.

الخلايا المختلطة

تجمع الخلايا المختلطة بين خواص الخلايا الديناميكية وخلايا المجموعات الأساسية. فهي تشبه خلايا المجموعات الأساسية في أن لها مجموعة أساسية لتخزين المفاتيح الفريدة من مجموعة النتائج، لكن لمجموعة الأساسية هي فقط مجموعة فرعية من مجموعة النتائج الكاملة. بمعنى آخر، تعلم الخلية بشأن بعض الصفوف في مجموعة النتائج وليس كلها. إذا قمت بتوجيهها لسحب بيانات من صف ليس من هذه المجموعة الأساسية المحدودة، فسيكون عليها سحب الصف بطريقة ديناميكية وهو ما ليس متاحا مع أوراكل كما قلنا من قبل.

خلايا Firehose

يجب إيضاح نقطة أخرى أخيرة في هذا القسم عن الخلايا مع بعض نماذج الوصول للبيانات (خاصة RDO وADO)، ستجد صورة بديلة أخرى للخلية، وهي خلية "للأمام فقط". تستخدم هذه الخلية عادة في طور للقراءة فقط لكنها قد تكون أحيانا قابلة للتحديث. وبدقة أكثر، ليست خلية "للأمام فقط" خلية على الإطلاق لكنها تحويل مباشر فقط لصفوف مجموعات النتائج من خادم أوراكل للمزود. ومن السهل تنفيذ خلية للأمام فقط وهي تستخدم الموارد بشكل كاف. على الرغم من أنك تبدأ مع الخلايا القابلة للتحديث للتطبيقات الصغيرة، فكلما زاد العبء على قاعدة البيانات كلما سيكون عليك الاعتماد على هذه الخلية البسيطة. من المناسب استخدام الخلية "للأمام فقط" لجميع أعمال التطوير الجديدة. ستحتاج القليل لتعاد عليها وسيكون عليك تصميم تطبيقاتك بشكل مختلف، لكنك ستحصل منها على النتيجة الموجودة. سنرى هذه الطريقة في الفصل الخاص بأساليب الوصول للبيانات مع بداية فصل ١٤ "وحدات الوصول إلى البيانات".

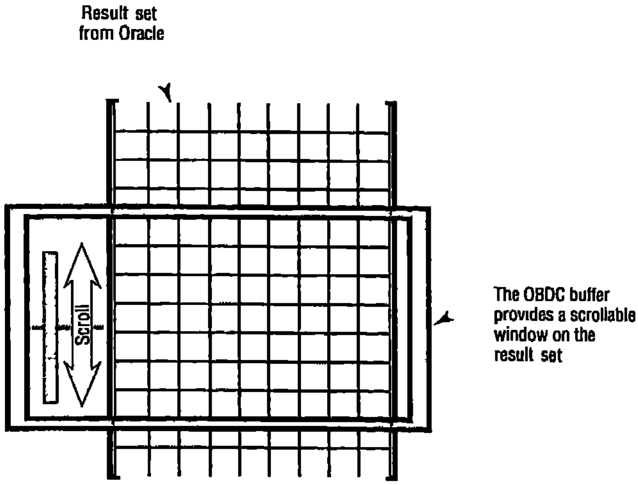
كيف تعمل الخلايا

أيا كان نوع الخلية الذي قررت استخدامه، فمن المفيد أن يكون لديك فهم عام لطريقة عملها جميعا لن ذلك قد يساعدك في تقدير نقاط أداء استخدام الخلايا. وكما ذكرنا سابقا، تكون البيانات العائدة من خادم أوراكل هي خلية للأمام فقط. يتم تخزين الصفوف المسترجعة في ذاكرة وسيطة على المزود (راجع الشكل ١١-١١). يتم تكوين هذه الذاكرة الوسيطة والاحتفاظ بها بواسطة مدير الخلايا. تتم قراءة صفوف خالية من البيانات من الخادم إلى الذاكرة الوسيطة للخلية. إذا كانت هناك صفوف كافية في مجموعة النتائج، فسيتم ملء الذاكرة الوسيطة ولن تتم قراءة المزيد من الصفوف في ذلك الوقت. تقوم

الذاكرة الوسيطة بتشكيل نافذة للبيانات الموجودة عليها. عندما تقوم بالتمرير للأمام حتى نهاية الذاكـرة الوسيطة، تتم قراءة صف جديد أو أكثر من الخادم. في نفس الوقت، تظل مجموعة النتائج على خـادم أوراكل في حالة معلقة "Pending" ولا يمكن إغلاقها حتى قراءة الصف الأخير، وهي بذلك تسـتهلك موارد النظام القيمة.

شكل ١١-١١

نموذج الخليـة يقرأ الصفـوف فـي ذاكــرة وسيطة.



إذن ما يحدث إذا أردت التمرير للخلف؟ ربما ستفقد البيانات ذات الصلة الموجودة فـي الذاكـرة الوسيطة، وسيكون من اللازم تنفيذ الاستعلام مرة أخرى. لن تلاحظ هذا لأن مدير الخلايـا سـيتدبر الأمر نيابة عنك. على أي حال، فهذه الخاصية هي أحد أسباب تفوق مجموعة سجلات للأمـام فقـط كأسلوب فعال من فيجوال بيسك على الخلية القابلة للإزاحة. إذا كنت تستخدم خلية مجموعة أساسـية ولا توجد ذاكرة كافية تضم المجموعة الأساسية، فسيتم نقل جزء منها إلى القرص.

نقطة أخيرة جديرة بالملاحظة هنا. لأن مشغل ODBC يدير الخلية، فإذا استخدمت خيار التمريـر حتى تستطيع تعليمات خاصة بأوراكل. فستتجاوز مدير الخلايا وستصبح مجموعة السجلات الناتجـة للقراءة فقط.

اختيار خلية

سيكون عليك دائما اختيار نوع خلية وخاصة لوحدات البيانات البعيـدة (RDO) ووحـدات البيانـات

النشطة (ADO). وكما ترى من الملاحظات بالأعلى، فإن لديك خيارا واحدا لخلية قابلة للتحديث مع أوراكل هي خلية المجموعة الأساسية. على أي حال، مع ADO، تستطيع الوصول فقط لخلية "للأمام فقط". وكما في نقاط كثيرة لاستخدام أوراكل من فيجوال بيسك، لا يكون لديك خيارات كثيرة. فقط تأكد من أنها أيهما هو اختيارك. ستعطيك خلايا القراءة فقط تصميما أفضل وأكثر قابلية للتحجيم، أما الخلايا الثابتة فستعطيك جميع الخصائص التي تتوقعها مثل القدرة على التحديث. أما الأفضل أداء، استخدم خلية للقراءة فقط للأمام فقط.

الخلايا في حيز التنفيذ

عندما تقوم بتكوين مجموعة سجلات قابلة للتحديث (ما كان يطلق عليه سابقا المجموعة الحيوية)، يقرأ المشغل فقط القيم الأساسية فقط للصفوف. (تقرأ وحدات OLE في ROWID معرف الصفوف لأوراكل). عندما تريد استرجاع الصف بالكامل، يقرأ مدير الخلايا الأعمدة المناسبة. عندما تقوم بتحديث أو حذف صف، يؤثر مدير الخلايا بهذا التغيير في قاعدة البيانات. (في حالة OLE-DB يتم تنفيذ هذا التغيير من خلال خدمة التزامن). بالطبع، كخلية نسخ نسبيا، فلن تخطر خلية المجموعة الأساسية التطبيق بأي تغييرات قام بها مستخدمون آخرون في السجلات في مجموعة السجلات حتى تقوم أنت بتحديثها. قد ينتهي بك الحال إلى محاولة تحديث صف قام أحدهم بحذفه للتو. عند التطبيق، تكون مشكلة محتملة وشيئا يجب أن تستعد له.

عندما تقوم بتكوين مجموعة سجلات للقراءة فقط (ما كان يطلق عليه مصطلح لقطة Snapshot)، تتم قراءة جميع البيانات في ذاكرة جانب المزود (في حدود عملية). ستجد عادة أن هذا النوع من مجموعات السجلات يكون أسرع كثيرا من النوع القابل للتحديث، على الرغم من أن هذا يعتمد على حجم مجموعة السجلات. بالطبع يمكن أن تكون متطلبات الذاكرة كثيرة، ومع مجموعة سجلات كبيرة إلى حد ما، ربما تجد أن المشغل يجب أن يعيد قراءة بعض أجزاء البيانات مع تأثير محتمل لأداء بطيء. لهذا السبب، يجب أن تحاول أن تتجنب بيانات من نوع BLOB في مجموعة سجلات للقراءة فقط.

تحذير: لتعمل مجموعات السجلات القابلة للتحديث في فيجوال بيسك، يجب أن تكون الصفوف قابلة للوصول بالمفاتيح. إذا لم يكن لديك مفتاح فريد أو أساسي على جدول، فلن يكون المشغل قادرا على إيجاد الصف الذي يبحث عنه، وستصبح مجموعة السجلات للقراءة فقط.

الوصلات

فيميل إلى التعامل مع وصلة قاعدة البيانات لربط قاعدة بيانات، ولكن في الواقع، فإن هذه نقطة يجب أن توليها بعضا من الاهتمام، خاصة وأنت تقوم بتطوير تطبيقات يستخدمها الكثيرون. وقد أصبحت القابلية للتحجيم أكثر أهمية. وطبقا للمورد، فإن أوراكل ٨ باستطاعته دعم وصلات متوافقة حتى ١٠,٠٠٠ وصلة أو أكثر، لكن هناك بعض الإصدرات والخصائص لقاعدة البيانات يجب أن تعرفها.

الوصلات والدورات

هناك مصطلحان يجب أن تستطيع التمييز بينهما هما: الوصلة والدورة. الوصلة هي طريق الاتصال بين إجراء الخادم الذي قام أوراكل بتكوينه وقاعدة البيانات نفسها. على الجانب الآخر، فالدورة هي الاتصال بين التطبيق البعيد (مثل برنامج فيجوال بيسك) وقاعدة البيانات التي تستخدم وصلة معينة.

والمصطلحات هنا مسببة للمشاكل لأن ما تسميه مايكروسوفت وصلة تطلق عليه أوراكل دورة. تستخدم وحدات أوراكل لـ OLE مصطلح "وصل" عندما تقوم بإنشاء دورة (راجع فصل ١٨). على أي حال، لأننا نعمل على بنية مايكروسوفت، فإن استخدام مصطلح وصلة لكلا الحالتين لا يبدو مسببا لأي مشاكل وسنستمر في هذا في الفصول التالية.

الوصلات المتعددة

إذا راقبت الإجراءات على خادمك، فستلاحظ أن هناك غالبا أكثر من وصلة لكل مستخدم. وهذا لأن طرق الوصول للبيانات اقتصادية نسبيا عند العمل مع أوراكل لكنها تتطلب وصلة منفصلة لتنفيذ SQL أو الإجراءات المخزنة. يدعم أوراكل شيئا يدعي مجموعات النتائج المعلقة وهي تعني أن نفس الوصلة يمكن أن تدعم مجموعات نتائج عديدة لم يتم إغلاقها، بالرغم من تفصيل تجنب ذلك. إذا كنت مصرا على استخدام هذه الإمكانية، فستكون في موقف أفضل مع أوراكل بالمقارنة بقواعد البيانات الأخرى مثل خادم SQL والذي تمكن أن تطلب منك الكثير من الوصلات، خاصة إذا كانت مجموعة النتائج كبيرة.

أحد أسوأ مشاكل استخدام الاتصال هي التحكم في البيانات وهي خاصية تتطلب افتراضيا وصلة منفردة لكل تحكم. وهذه الوظيفة مضيعة تماما للموارد (لكل وصلة مساحة SQL الخاصة بها ذاكرة الخادم)، وهي تعني أنه قد يمر وقت طويل حتى تستطيع الاتصال. لحسن الحظ، هناك أسلوب للمشاركة في الوصلات بين وظائف التحكم في البيانات، وهي ما سنراه في فصل ١٤، وحدات الوصول للبيانات.

ملاحظة

يتم الترخيص لأوراكل حسب المستخدمين وليس الوصلات. كنتيجة لذلك، لن يكون لديك مشاكل تتعلق بالترخيص لوصلات متعددة لتطبيق. على أي حال، إذا استخدمت الخادم المتعدد سلاسل العمليات، فسيكون هناك مستخدمون أكثر من الوصلات.

وصلات DSN-Less

رأينا كيف نضع سلسلة وصلات معا خلال الأقسام السابقة. هناك تفاصيل عميقة لهذه الطريقة مثل لزوم إنشاء مصدر البيانات في برنامج مسئول ODBC لكل حاسب شخصي. لهذه الطريقة أيضا إصابة صغيرة على التسجيل Registry عند قراءة المعلومات ذات الصلة. على أي حال، ليست هذه هي الطريقة الوحيدة لإنشاء الوصلة. تستطيع الآن تجاوز هذا الإجراء باستخدام وصلة DSN-Less. يتم تحقيق هذا بتحديد جميع المعلومات المطلوبة بما فيها اسم مشغل ODBC عندما تقوم بالاتصال. سنرى هذا في فصل ١٦، ODBC Direct والفصول التالية.

إغلاق وصلة ODBC

ربما ستفاجأ عندما تستخدم محرك JET للوصول إلى أوراكل (أو أي قاعدة بيانات ODBC أخرى بهذا الخصوص) وتجد من الصعب إغلاق وصلة. يمكن أن تكون هذه الخطوة واضحة عندما تشغل تطبيقا لفيجوال بيسك يطلب منك اسم المستخدم الخاص بك مع كلمة السر. إذا أوقفت التطبيق وأعدت تشغيله، لن تتم مطالبتك بشيء مرة أخرى. ما يحدث هو أن JET سيحتفظ بالوصلة مفتوحة حتى لو فعلت كل ما تتخيل لإغلاق قاعدة البيانات. وهو يفعل هذا حتى لا يعيد الاتصال بشكل متكرر. يمكن أن تصبح هذه مشكلة إذا قمت بالتحول بين اختبار وإنتاج وتحويل اسم الخدمة في تحديد DSN.

إذا أردت التأكد من إغلاق الوصلة، فسيكون عليك الانتظار لمدة الثواني المحددة في معامل

Connection Timeout في التسجيل Registry. القيمة الافتراضية هي ٦٠٠ ثانية. على أي حال، تستطيع تغيير هذه القيمة الافتراضية بتغيير قيمة معامل Connection Timeout إلى قيمة صغير مثل ١. البديل السريع هو إعادة تشغيل بيئة فيجوال بيسك بسرعة.

أحذر من أية إجراءات تغييرات على Registry. إذا لم تكن متأكدا من هذا التغيير، قم بنقل Registry إلى ملف قبل تغييره لأنك إذا قمت بشيء خاطئ فستجد أن من المستحيل إصلاحه. أنا شخصيا لا أحبذ اللجوء إلى التغيير اليدوي لـ Registry، حيث يمكن لهذا أن يكون لديك الكثير من أجهزة المزودين١. إذا كان يجب عليك فعل ذلك، فعليك التفكير في كتابة برنامج للقيام بهذه التغييرات بدلا من إجراءات لها يدويا.

تجميع وصلات ODBC

إحدى الإضافات الجديدة لإمكانيات ODBC تسمى بتجميع الوصلات. وهي قدرة على مدير مشغلات ODBC على المشاركة تبين وصلات ODBC طالما كانت هذه الوصلات تستخدم نفس مشغلات ODBC. جاءت هذه الخاصية مع الإصدارة ٣ من مدير مشغلات ODBC وما بعدها وهي مستقلة عن المشغل الفعلي.

تكون القدرة على المشاركة بين وصلات قواعد البيانات شديدة الأهمية عندما يكون لديك بنية مركبات يكون فيها عدد كبير من المركبات على جهاز واحد تنتظر لتقوم بإنشاء وصلة بشكل متكرر لنفس قاعدة البيانات. (لا يمكن أن تشترك في وصلة في نفس الوقت). وبسبب صعوبة إنشاء وصلة بأوراكل (وقواعد بيانات أخرى)، فيجب أن تكون هناك طريقة لتجنب البداية والإغلاق المستمر لإجراءات الخادم.

إذا كنت تستخدم خادم مايكروسوفت للحركة (MTS)، فلا تقلق بشأن تجميع الوصلات لأن هذه الخاصية تكون مدمجة فيه. على أي حال، فإن دعم أوراكل حتى الآن لـ MTS ليس مشجعا حيث يدعم أوراكل بنية مشابهة إلى حد ما مع خادم تطبيقات أوراكل.

إذا كنت تريد متابعة هذا، فهناك مقالتان قاعدة معرفة مايكروسوفت قد ترغب في مراجعتهما (Q169470 و Q164221). قد ترغب أيضا في تبين ما إذا كانت هناك فائدة في خادم أوراكل المتعدد سلاسل العمليات (الذي يحتوي على مكافئ MTS) أو الوصلات البادئة سابقا Prestarted Connections، الذين سنناقشهما في الأقسام التالية.

خادم أوراكل المتعدد سلاسل العمليات

عندما تبدأ العمل مع أوراكل، فمن المحتمل أن تستخدم بنية الخادم المكرس التي رأيناها في فصل ٢. على أي حال، رأينا في فصل ٣ التكوين الذي تحتاجه لتشغيل إجراءات الإرسال المستخدمة في خادم أوراكل المتعدد سلاسل العمليات. تسمح لك البنية بمشاركة وصلة بين عدة مستخدمين وهي سبب هام في إمكانية أوراكل دعم هذا العدد الكبير من الدورات المتوافقة. عند التطبيق، فإن تشغيل MTS كان صعبا، ويجب أن تعتبره خاصية متقدمة. إصدارة MTS في أوراكل ٨ قد تطورت كثيرا. على أي حال، يجب عليك أن تفكر في استخدامه عندما تحتاج المئات من الوصلات المتوافقة أو أكثر. يفضل MTS فقط لاستخدام OLTP.

إذا كنت تعمل مع عدد كبير من الوصلات المتوافقة، فقد ترغب أيضا في استكشاف خاصية في أوراكل ٨ تسمى مدير الوصلات التي تسمح بالمضاعفة التي تشبه تجميع وصلات ODBC.

الوصلات التي بدأت من قبل

إذا لم تكن تستخدم خادم أوراكل المتعدد سلاسل العمليات، فعند إجراء لطلب وصلة سيبدأ إجراء المنتصت إجراء خادم مكرس جديد. يمكن لهذا أن ستغرق ثانية أو نحوها ليتم تنفيذه، ويمكن لأوراكل أن ينشئ لك وصلات بدأت من قبل.

تستطيع إعداد وصلات بدأت من قبل بتزويد معاملات إضافية في ملف Listener.ora. عند بداية إجراء المنتصت على قاعدة البيانات، يتم إنشاء عدد معين من الوصلات لكنها لا تستخدم من أي دورات. عندما تبدأ تطبيق فيجوال بيسك، سيبحث المنتصت عن وصلة متاحة "غير مستعملة" من التجميع. وعند انتهائه من الوصلة، سيتم إرجاعها إلى التجميع.

يعتمد التأثير على تطبيق فيجوال بيسك على التقنية التي تستخدمها لكنها لن تكون ذات تأثير كبير كما تتخيل. يجب فقط أن تفكر في هذا الأسلوب عندما يكون لديك عدد كبير من الوصلات المتزامنة، على سبيل المثال، في تطبيق إنترنت.

تلميح قبل التفكير في استخدام الوصلات المبدوءة سابقا، يجب أن تتأكد أن وقت وصلتك بطيء. لاختبار وقت الوصلة يجب أن تستخدم SQL*Plus على الخادم. لا تستخدم برنامج فيجوال بيسك لأن ذلك شديد التعقيد. على سبيل المثال، مع وحدات الوصول للبيانات، جميع بيانات الجداول المتاحة يتم تحميلها للمزود ثم يمكن إظهار الوصلة لتأخذ وقتا أطول مما تأخذه في الحقيقة.

طرق وصول بديلة

لا شك أن أساليب مايكروسوفت بما فيها ODBC و OLE-DB تتحكم في التعامل بين أوراكل وفيجوال بيسك. على أي حال، هناك بعض الطرق الأخرى التي يمكن استخدامها وهي مترجم أوراكل الأولى ووحدات أوراكل لـ OLE. على الرغم من أن مترجم أوراكل الأولى لن يجذب الكثير من مبرمجي فيجوال بيسك، يجب أن تعرف وحدات أوراكل لـ OLE لأنها تتيح لك الاستفادة بأقصى فاعلية لقاعدة البيانات.

مترجمو أوراكل الأوليون

من الممكن كتابة تطبيقات تصل إلى OCI API باستخدام استدعاءات المستوى المنخفض المتاحة. هذا الإجراء يشبه الكتابة لـ ODBC API. ربما ما لا يعلمه مطورو فيجوال بيسك هو أن لأوراكل خمسة مترجمين أولين للغات البرمجة تعمل على العديد من أنظمة الحاسبات. (يعالج المترجم الأولى تعليماتك قبل تمريرها إلى المترجمين). يسمح لك المترجمون الأوليون باستخدام SQL متضمنة التي يأخذ فيها المترخم الأولي تعليماتك ويحولها SQL المتضمنة إلى استدعاءات مباشرة لواجهة تطبيق استدعاء أوراكل. المترجمون الأوليون الموجودون هم: Pro*C/C++ و Pro*Cobol و Pro*Fortran و Pro*ADA و Pro*PL/1 يتاح على أنظمة حاسبات أقل. وعلى الرغم من أن OCI متاحة لنفس اللغات، فإن استخدام المترجمين الأوليين يكون أسهل كثيرا من الكتابة لـ OCI.

ملاحظة تستطيع نظريا الوصول إلى OCI من فيجوال بيسك. على أي حال، فإن هياكل البيانات صعبة البرمجة، كما أن مجموعة VB يمكن أن تجعل ذلك غير موثوق به أو مستحيلا. لا أعتقد أن أحد قد يجرب ذلك.

وحدات أوراكل لـ OLE

البديل الأخير لاستخدام ODBC هو منتج من أوراكل لعدة سنوات يسمى وحدات أوراكل OLE (OO4O) قد يسبب الاسم بعض المشاكل لأن OLE قد تذكرنا بـ COM (نموذج وحدات المركبات). OO4O هي dLL تعمل في نفس إجراء تطبيق فيجوال بيسك الخاص بك وتوفر ربطا مباشرا بواجهة تطبيق استدعاء أوراكل. من السهل إتقان OLE على أي مستخدم عمل كثيرا على برمجة قواعد البيانات في فيجوال بيسك لأنها تستخدم نفس نموذج الوحدات مثل DAO و RDO المعروفين. سنرى

المزيد عن OO4O في فصل ١٨ "وحدات أوراكل لـ OLE". يوفر الخادم العام ربطا مباشرا بواجهة تطبيق استدعاء أوراكل ويجب أن يوفر الخصائص، مثل الخلايا، التي لا توجد في OCI.

خلاصة

غطى هذا الفصل الخلفية التي يجب أن تكون لديك لتفهم كيفية اتصال تطبيقات فيجوال بيسك الخاصة بأوراكل. ناقشنا كلا من برامج أوراكل للشبكات وبنية مايكروسوفت ODBC بتفصيل كاف لتركيب وتكوين واختبار كليهما. ناقشنا أيضا OLE-DB وهي أحدث تقنية واعدة فيما يتعلق بالفعالية والأداء والديمومة العامة. هذه المعلومات هامة أيضا لك لتكون قادرا على إنتاج قناة اتصال فعالة بين فيجوال بيسك وقاعدة البيانات.

وبناء على هذا الأساس، نستطيع التحرك الآن إلى الطرق المتنوعة التي يمكنك بواسطتها وكمطور فيجوال بيسك، الوصول إلى البيانات في قواعد بيانات أوراكل الخاصة بك.

١٢

الوصول لأوراكل من خلال فيجوال بيسك

هناك طرق متعددة للوصول إلى قاعدة بيانات أوراكل من فيجوال بيسك، ولكل طريقة ميزاتـــها وعيوبها. على الرغم من أنك تستطيع اختيار أنسب الطرق للوصول حسب حاجتك، فيجب عليـك أولاً أن تكون على علم بهذه الطرق المتنوعة. إذا كانت هناك طريقتين أو ثلاثة لكان من السـهل التعرف عليها جميعاً. ولكن في الحقيقة هناك الكثير من الطرق التي يمكنك الاختيار من بينـها، وتزيد هذه الطرق عاماً بعد آخر. ما هو أصعب من ذلك هو أن التفصيل والإمكانيات لكل طريقة تبدو وكأنها تتغير كل بضعة أشهر، يأتي هذا كنتيجة للتنافس الشديد في هذا المجال.

سيعطيك هذا الفصل نظرة شاملة على الطرق المتنوعة التـي يمكنـك اسـتخدامها حاليـاً. وسأستطرد قليلاً وأخمن عدد من الطرق التي ستستقر لعدة سنوات بما أن لدينا الآن الـــ OLE-DB القياسية من مايكروسوفت. على أي حال، قبل أن نبتهج أكثر من اللازم، دعني أضيف هنـا أن تفاصيل معظم هذه الطرق تتغير تقريباً بمعدل ثابت كل فترة.

في الأقسام القليلة التالية، سنأخذ فكرة عامة على الطـرق التاليـة مـع مقارنـة الأسـاليب والخصائص لكل طريقة. والموضوعات المتنوعة التي سنناقشها هي:

- محرك قواعد بيانات JET
- وحدات البيانات البعيدة
- ODBC Direct
- ODBC API
- بيانات Active X
- وحدات أوراكل لـ OLE
- تصدير قاعدة بيانات أكسس لأوراكل

لكل نوع من طرق الوصول إلي البيانات، سنراجع الخصائص بالإضافة إلي تبيـن أجـزاء التعليمات البرمجية التي يمكننا من خلالها معرفة كيفية استخدام كل أسـلوب للاتصـال بقـاعدة بيانات لا وراكل وإنشاء مجموعة سجلات. وقد تم إعداد مؤشر مجموعة السجلات الأسرع فـي كل حالة لان هذه هي الطريقة المفضلة لاسترجاع البيانات من قاعدة بيانـات أو راكـل. سـيتم مناقشة معظم الأساليب بتفصيل أكثر خلال بقية الكتاب لكن هذا الفصل سيعطيك تلخيصا سـريعا ليساعدك لتقرر أكثر الأساليب فائدة لك.

محرك JET

كان JET هو البداية، فقد كان فيجوال بيسك ٣ خطوة متقدمة للتعامل مع قواعد البيانات. وقد وفر طريقة سهلة للوصول إلي بيانات قواعد بيانات أكسس وقواعـد بيانـات ISAM أخـري مثـل FoxPro. للوصول إلي قواعد البيانات هذه ستستخدم محـرك بيانـات JET (Joint Engine

Technology). يعتبر معظم المبرمجين خاطئين أن هذا المحرك جزء من فيجوال بيسك لكـــن الحقيقة هي أن ملفات التشغيل اللازمة JET تأتي مع VB منا تأتي ملفات أخري كثيرة، فلا يوجـد تكامل معين VB وJET. إذا لم تقم بإرفاق المراجع الصحيحة في مشروع فيجوال بيسك الخـاص بك، فلن يتعرض تطبيقك على أي عنصر من عناصر قاعدة البيانات.

لاستخدام محرك قاعدة بيانات JET، توفر عناصر الوصول للبيانات (DAO) طريقة تعتمـــد على الوحدات تتمثل في سلسلة من الوحدات كقاعدة البيانات نفسها، و مجموعة مـــن الســـجلات المسترجعة تسمى مجموعات السجلات (Record sets). تعرض مجموعة السجلات الخصـائص والطرق التي تسمح لنا بتجديد تفاصيل الارتباط بالوحدات بالإضافـــة إلـــي تصفـح الســجلات وتحديثها. لمحرك JET معالجات الاستعلامات ومجموعة النتائج الخاصة به بحيث يستطيع قـراءة البيانات من ملفات قواعد البيانات ويعرف كيف يتعامل مع تأمين الملفات عندما يكون هناك أكـثر من مستخدم. على أي حال، يتسبب معالج الاستعلامات أيضا في زيادة حجم البرنامج مـــع عــدم القدرة على التعامل SQL بشكل سليم بالإضافة إلي إبطاء سرعة العمل. ما هو أكثر من ذلك هـــو أن JET لا يستطيع الاستفادة من الخصائص المتطورة لقاعدة بيانات نشطة بعيدة يمكن أن يكـــون على الاتصال بها (مثل أوراكل). لاستخدام هذه الخصائص المتطورة، يجب عليك تجاوز محـــرك JET الذي سنقوم بمناقشته في فصول تالية.

على أي حال، فإن محرك JET يكون مهما في تزويد هيكل للوحـــدات يعتمـد عليــه بقيــة الوحدات وعلى الرغم من هذا وكما سنرى، من تغيير أسماء الخصائص لأسباب غير واضحـــة توجد طريقة حديثه مع ADO وهي تبسيط عملية الوصول إلي البيانات نسبيا بالإضافة إلي تحسين أدائها لتكون قادرة على الوصول إلي البيانات من جميع الموارد الممكنـــة وليـــس فقـــط قواعــد البيانات.

الكثير من مبرمجي فيجوال بيسك إن لم يكن كلهم يبدءون مع قواعد بيانات أكســـس. يوفـــر فيجوال بيسك. الوظيفة الإضافية Data Manager ليسمح لنا بإنشاء قواعد البيانات دون شـــراء برنامج أكسس. سوف تفيد معلوماتك عن أكسس مباشرة لتنتقل إلى قاعدة بيانات أكـــثر تطـــورا كأوراكل مثلا على أي حال، يجب عليك أن لا تجعل خبرتك في أكسس تحد تفكـــيرك فالانتقـــال بسرعة لقاعدة بيانات خارجية باستخدام ODBC ليس بالمهمة الصعبة كما قد تتصور. فعند طريق اتباع الخطوات البسيطة الموضحة في هذا الكتاب، سيكون باستطاعتك الانتقـــال إلـــى أوراكـــل بسهولة شديدة. ولتصبح خبيرا في ذلك فستحتاج طبعا إلى المزيد من الوقت والتجريب.

على الرغم من أنك قد لا تستخدم JET كثيرا في تطبيقات إنتاج، ينبغي أن تكون علـــى علـــم بالتقنيات التالية:

- ربط الجداول بقاعدة بيانات أكسس

- استخدام عناصر تحكم البيانات
- استخدام وحدات الوصول إلى البيانات

الجداول المرتبطة

من أسهل الطرق لبداية فيجوال بيسك وأوراكل استخدام الجداول المرتبطة (التي كانت تسمى سابقا الجداول المرفقة Attached Tables). واستخدام الجداول المرتبطة هي طريقة أقرب للأعداد أكثر من كونها طريقة للوصول مما يجعل جداول أوراكل متاحة مع أساليب DAO القياسية. (سنناقش أساليب DAO في القسم التالي) تستطيع أيضاً استخدام النماذج والاستعلامات الخاصة بتطبيق أكسس نفسه كبديل لاستخدام فيجوال بيسك.

لاستخدام الجداول المرتبطة، نقوم بإنشاء قاعدة بيانات أكسس ونستخدم إمكانيات أكسس لربط أو إرفاق جدول من مصدر خارجي مثل أوراكل بجدول أكسس. ما يحدث في الواقع هو دمج بناء قاعدة بيانات أوراكل في جدول أكسس بما يتيح لمحرك JET معرفة كيفية قراءة والتعامل مع البيانات. يقوم محرك JET بترجمة أية مطالب وإرسالها إلى أوراكل. توفر هذه العملية قدرا كبيرا من الكفاءة لأنه عند استخدامك لأسلوب وصول للبيانات مثل وحدات وصول البيانات DAO يتم تخزين جميع تسميات الجدول في الذاكرة. وكون هذه التسميات في قاعدة بيانات أكسس يوفر الوقت ومصادر الخادم "طالما التعريفات لا تتغير".

يمكنك معرفة هيكل جدول أوراكل في قاعدة بيانات أكسس من ربط جداول من مصادر مختلفة مثل أوراكل وخادم SQL لمايكروسوفت وSybase وتشغيل استعلامات عنها جميعا في نفس الوقت. وعلى الرغم من أن هذا يبدو جيدا، فهذه طريقة لا نوصى بها تماما لأنه لها نقاط ضعف مؤثرة:

- بالإضافة إلى انعدام أداء JET مع قواعد البيانات البعيدة فيمكن أن تأخذ جداولك المرتبطة موارد معقولة من موقع عمل مزودك
- هناك قلة في الخصائص كتنفيذ الإجراءات المخزنة
- تحتاج الجداول المرتبطة للإعداد على كل جهاز حاسب لكل مستخدم ويجب تحديثها عند أحداث أي تغييرات في هياكل قواعد البيانات
- يتم تقييدك بصيغة SQL غير القياسية لأكسس

ربط جدول من أوراكل بأكسس

أول ما سنقوم به هو ربط جدول من أوراكل بقاعدة بيانات لأكسس سيستخدم هنا أكسس ٩٧ لكن التقنيات تعمل أيضا إذا كنت لا تزال تستخدم إصدارة أكسس ٢ ذات ١٦ بيت وتكون الخطوات مباشرة تماما:

١- قم بإنشاء قاعدة بيانات أكسس(ملف Mdb).

٢- اختر خيار القائمة File>External Data>Link Tables.

٣- في أسفل مربع حوار Link في مربع التحرير والسرد "File OF Type"، اختر "ODBC Data Bases".

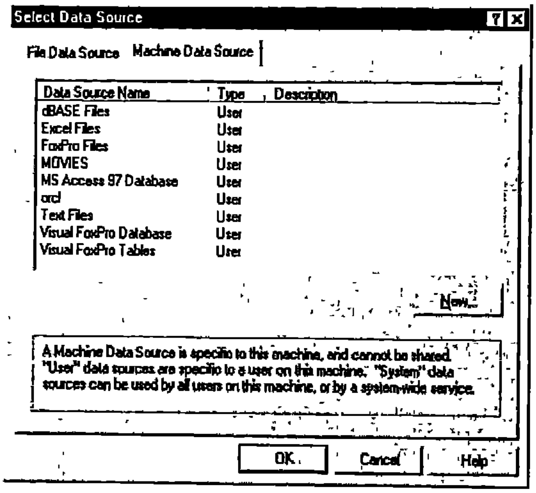
٤- حدد اسم جدول أوراكل الذي تريد ربطة (انظر الشكل ١٢ - ١). ستكون هذه الخطوة تحت علامة جدولة Machine Data Source.

٥- قم بتسجيل الدخول باسم المستخدم الخاص بك في أوراكل مع كلمة السر.

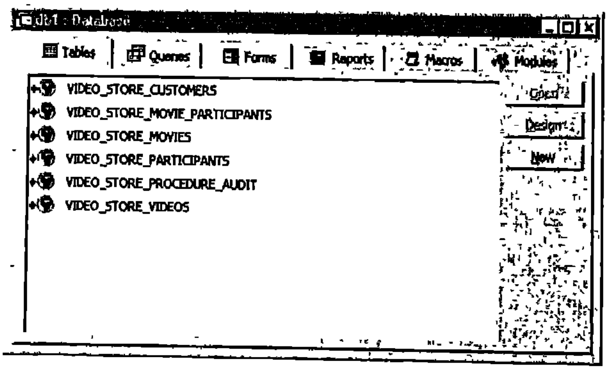
٦- اختر جدولا أو أكثر لتقوم بإرفاقه بقاعدة بيانات أكسس.

٧- أغلق مربع الحوار.

سيظهر الجدول في نافذة قاعدة بيانات أكسس كما هو موضح في الشكل ١٢-٢



الشكل ١٢-١
اختيار مصدر بيانات لإرفاقه بقاعدة بيانات أكسس الخاصة بك.



الشكل ١٢-٢
قاعدة بيانات أكسس بها ستة جداول مرفقة أوراكل

ملاحظة لإزالة الجدول المرفق، قم بتحديده وانقر مفتاح الحذف واستخدم خيار القائمة Edit؛Delete.

نستطيع الآن استخدام الجداول المرتبطة في أكسس كما نفعل مع أي قاعدة بيانات أخري بما فيها محرك JET لأكسس أو نستطيع استخدام فيجوال بيسك للوصول إلى الجداول المرتبطة كما لو كانت قاعدة بيانات لأكسس.

تكون قيود الجداول المرتبطة دائمة بحيث لا تستطيع استخدام أية خصائص معينة لأوراكل كالدوال أو الإجراءات المخزنة وهو ما يمكن أن يكون عائقا كبيراً، كما أن الأداء الأقل كفاءة يمثل مشكلة كبيرة، ومع الأساليب المتاحة الآن، لماذا تستخدم محرك JET من البداية، فلا توجد له مميزات لبيئة الإنتاج مع أنه مع عناصر تحكم البيانات، يمكن استخدامها لإعداد النماذج أو كأدوات إدارية لن يتم استخدامها كثيرا.

تعليمات برمجية نموذجية

عندما تستخدم الجداول المرتبطة فإنك تتعامل مع أكسس طالما أنت تعمل من خلاله. وكنتيجة لذلك، فإن التعليمات البرمجية النموذجية التي تراها في التعليمات البرمجية ١٢-١ ستكون مطابقة تقريبا لما كنت ستكتبه إذا كنت تعمل في قاعدة بيانات أكسس عادية باستخدام DAO. اسم مصدر البيانات Data Source Name (DSN)، في هذه حالة "Movies" يشير إلى ملف قاعدة بيانات أكسس Movies Mdb تنشى التعليمات البرمجية وحدة قاعدة بيانات ثم تفتح مجموعة سجلات لها.

تعليمات برمجة ١٢-١

```
Private Sub FetchAttached()
   Dim dbMovies As Database
   Dim rsMovies As Recordset
   Dim strSQL As String
   Dim iField As Integer
'  Connect to the database
   Set dbMovies = Workspaces(0).OpenDatabase("", False, False, _
     "DSN=movies;UID=video_user;PWD=ummagumma")
'  Run the query
```

```
    strSQL = "SELECT * FROM movies ORDER BY movie_id"
    Set rsMovies = dbMovies.OpenRecordset(strSQL, dbOpenSnapshot)

'   Display the fields
    While Not rsMovies.EOF
        For iField = 0 To rsMovies.Fields.Count – 1
            Debug.Print rsMovies(iField)
        Next ' iField
        rsMovies.MoveNext
    Wend

    Set rsMovies = Nothing
    Set dbMovies = Nothing

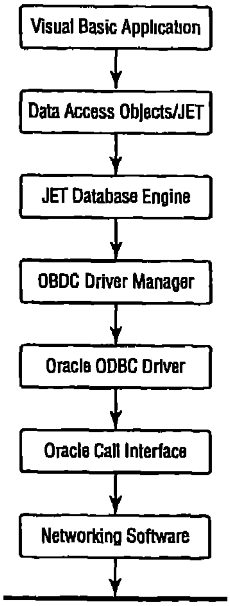
End Sub
```

وحدات الوصول للبيانات

وحدات الوصول للبيانات (DAO) Data Access Objects هي عماد إمكانيات قواعد بيانات فيجوال بيسك منذ فترة طويلة. في هذه الإصدارات ٣,٥١ لدينا منتج مختلف عند البداية تماماً. DAO هي التقنية الرئيسية المستخدمة في التعامل مع قواعد بيانات أكسس من خلال ملفات تشغيل محرك JET لكنها أيضا قادرة على التعامل مع مصادر البيانات الداخلية الأخرى مثل ملفات إكسل. بالإضافة إلى ذلك، فقد أضافت مايكروسوفت القدرة على ربط مصادر بيانات ODBC وهى الخاصية التي كان المستخدمون يطلبونها كثيراً. يعطي الشكل ١٢-٣ فكرة عامة عن طبقات البرامج المستخدمة في هذه الطريقة. سنناقش DAO بتفصيل اكبر في فصل ١٤ وحدات الوصول لبيانات.

الشكل ١٢-٣

طبقات البرامج المستخدمة للوصول إلى أوراكل من خلال DAO.



قد تكون DAO أحد أهم أسباب شهرة ODBC، بالبديل الوحيد في الوقت الحالي هو برمجة لوحة برمجة التطبيق (API) التي كما سنرى لاحقا عبارة عن تقنية معقدة ومضيعة للوقت بحيث لا يتحمس لها مبرمجون كثيرون ولا يجد لها مديرو المشروعات موارد كافية.

خصائص

- تستطيع شراء Data Access Objects الإصدار ٣,٥١ الموجودة الآن على شكل ٣٢ بيت. على أي حال، فما زالت الإصدار ٦٦ بيت متاحة لمستخدمي فيجوال بيسك ٤ أو الإصدارات سابقة.
- تستطيع القراءة والكتابة في جداول قواعد البيانات.
- تستطيع الوصول إلى قواعد بيانات مختلفة في نفس الوقت أو حتى الجداول المبرمجة من قواعد بيانات مختلفة مثل أوراكل و JET وخادم لمايكروسوفت SQL.
- تستطيع استخدام SQL Pass Through لتتجاوز محرك JET وسيتم إرسال الاستعلامات مباشرة إلى أوراكل على الرغم من أن مجموعات السجلات المنشاة بهذه الطريقة تكون متاحة للقراءة فقط.
- تستطيع استدعاء الإجراءات لكنك لا تستطيع استرجاع أي معاملات إخراج.

تعليمات برمجية نموذجية

التعليمات التي تراها في التعليمات البرمجية ١٢-٢ تتطابق تقريبا مع تلك التي في التعليمات البرمجية ١٢-١ على أي حال، في هذا المثال فقد تم استخدام كلمة ODBC الأساسية في تسلسل الاتصال والآن يشير DSN لاسم قاعدة بيانات ODBC مسجلة.

تعليمات برمجية ١٢-٢

```
Private Sub FetchDAO()
   Dim dbMovies As Database
   Dim rsMovies As Recordset
   Dim strSQL As String
   Dim iField As Integer
' Connect to the database
   Set dbMovies = Workspaces(0).OpenDatabase("", False, False, _
     "ODBC;DSN=movies;UID=video_user;PWD=ummagumma")
' Run the query
   strSQL = "SELECT * FROM video_store.movies ORDER BY movie_id"
   Set rsMovies = dbMovies.OpenRecordset(strSQL, dbOpenSnapshot)
' Display the fields
   While Not rsMovies.EOF
      For iField = 0 To rsMovies.Fields.Count - 1
         Debug.Print rsMovies(iField)
      Next ' iField
      RsMovies.MoveNext
   Wend
Set rsMovies = Nothing
   Set dbMovies = Nothing
End Sub
```

عناصر التحكم في البيانات

ربما كان الشيء المشوق في فيجوال بيسك للمبتدئين هو التحكم في البيانات Data Control وعناصر التحكم المرتبطة. وعنصر التحكم للبيانات هو عنصر سحب وإفلات بسيط يسمح

بالوصول المباشر لجدول معين في قاعدة البيانات. تقوم أنت بسحب عنصر التحكم لتقوم بتشكيل وتحديد مصدر البيانات والجدول، ثم تنشئ ارتباطا لأي جدول في أوراكل في دقائق معدودة. وبمجرد إتمامك لهذه العملية، تستطيع إملاء العديد من عناصر التحكم الأخرى على نفس النموذج. تسمح لك عناصر التحكم هذه بدمج قيمها المعروضة في النموذج وفى كل مرة تستخدم فيها أزرار التصفح لتغيير السجل الحالي ستتغير أي عناصر تحكم مرتبطة على النموذج نتائجها المعروضة لتماثل السجل الحالي الجديد. إذا قمت بإدخال قيمة جديدة في عنصر التحكم المرتبط وانتقلت إلى السجل التالي، سيتم تحديث القيمة في قاعدة البيانات آلياً.

يعتبر هذا البرنامج بلا تعليمات برمجية.ولا يستطيع أحد إنكار مدى قبول البرامج غير ذات التعليمات للمبرمجين الذين مازالوا في البداية خاصة أولئك الذين في عجلة من أمرهم لإنتاج تعليمات برمجية بشكل محترف. ميزة أخري اكبر هي أن عناصر التحكم الجديدة والكثيرة تتعامل مع دمج البيانات بتطور اكبر عن مربع نص بسيط لتحديث حقل أو اثنين. على سبيل المثال، هناك عناصر تحكم متشابكة معقدة تسمح لك بعرض وتحديث سجلات من جداول مرتبطة، وهذه هي البرمجة في ابسط صورها. إذا كنت تتابع مجلات الكمبيوتر، سيكون لديك الانطباع أن هذه الطريقة الوحيدة المقبولة للبرمجة، فمن الذي يستخدم طريقة أخري للبرمجة ؟ فعناصر التحكم المرتبطة هي مقاييس شديدة الفنية وهى الطريقة الأفضل للعمل أو هكذا ستعتقد.

على الرغم من فائدة عناصر تحكم البيانات في مواقف العمل المشترك الصغيرة مع قاعدة بيانات أكسس إمكانية أن تكون نموذجية عندما تقوم بتكرار تطبيق، فعندما تكون مرتبطا مع قاعدة بيانات لمؤسسة (باتساع شركة بالكامل وحيث يكون أمان وكفاءة البيانات أمرا مهما)، لا يجب أن تستخدم عناصر التحكم المرتبطة.

عالم عناصر التحكم في البيانات

تستطيع استخدام الكثير من عناصر التحكم في البيانات لتنشئ اتصالا بقاعدة بيانات أوراكل. لدينا عناصر التحكم المألوفة المتوفرة مع كل إصدارة لفيجوال بيسك، ثم هناك عنصر التحكم في البيانات البعيدة الذي يأتي مع وحدات البيانات البعيدة. تحتوى أيضا وحدات أوراكل لـ OLE على بديل مشابه تماما " DRO pin " لعنصر البيانات. وتقوم وحدات بيانات ActiveX، وهي احدث التقنيات التي توفرها مايكروسوفت، بتطوير هذا المفهوم لتوفر عنصر التحكم في البيانات لخدمات البيانات البعيدة وهي التي تمكن أن توفر معاملا بغير اتصال على الإنترنت.

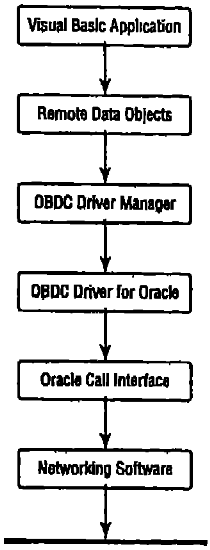
فيما عدا بعض التلميح المختصر في هذا الفصل، فقد تم استبعاد جميع أنواع عناصر التحكم في البيانات من هذا الكتاب وذلك لأسباب عديدة:

- عناصر التحكم في البيانات سهلة الاستخدام بوجه عام ويتم تفصيلها تماما في كتب أخري كثيرة.
- تتميز بيئة أوراكل بأنها يتم التحكم فيها بدقة ليتم تأمينها ولن تكون قادرا على استخدام خاصية عناصر التحكم في البيانات للتحديث. وبدون هذه الخاصية، ينتهي المغزى الرئيسي من استخدامها.
- أوراكل هو قاعدة بيانات high-end ولا يستطيع اغلب المطورين التحكم في تأمين البيانات أو حجبها مع عناصر التحكم في البيانات.
- لا يتم استخدام عناصر التحكم في البيانات عادة في بيئة متعددة الطبقات ولذلك فإن قابلية التحجيم لها محدودة.
- تعمل عناصر التحكم في البيانات بشكل عام ولا تدعم عناصر أو تكوين خاص بأوراكل (مع استثناء عنصر تحكم وحدات أوراكل (OLE).
- لا تستطيع عناصر التحكم تشغيل إجراءات مخزنة.
- لا تستفيد عناصر التحكم في البيانات كثيرا من موارد الخادم مثل اتصالات ODBC (أيضا مع استثناء عنصر) تحكم وحدات أوراكل (OLE).

وحدات البيانات البعيدة

مع استخدام DAO المتزايد، أصبح من الواضح أن مواجهة مشاكل الأداء وقلة خصائص DAO تمكن فقط بكتابة تعليمات برمجية بسيطة ODBC API. على أي حال، لم تكن هذه الطريقة هي المهمة الأسهل، فإذا لم تكتب تعليماتك البرمجية جيدا فإن ذلك سيؤدى إلى إنشاء تطبيق ضعيف و سيئ الأداء. لذا أنتجت مايكروسوفت وحدات البيانات البعيدة Remote Data Objects (RDO) وهى غلاف حول API التي تجمع العديد من خصائصها وتوفر محسنات الأداء التي كان يطالب بها مطورون كثيرون جاءت RDO مع الإصدار ١,٠ التي ظهرت مع إصدار Enterprise Edition من فيجوال بيسك ٤. ثم جاءت نسخة أكثر تطورا مع فيجوال بيسك ٥، على الرغم من إضافة تعديلات ثانوية فقط لإصدار فيجوال بيسك ٦. يعطى الشكل ١٢-٤ نظره شاملة على طبقات البرامج التي تستخدمها RDO للاتصال بقاعدة بيانات أوراكل. إذا قارنت هذا بالشكل ١٢-٣، سترى أن JET تم استبعاده تماما من الحلقة. سنقوم بمناقشة RDO بقدر أكبر من التفصيل في الفصل ١٥، وحدات البيانات البعيدة.

الشكل ١٢-٤
طبقات البرامج المستخدمة في وحدات البيانات البعيدة.



خصائص

- تتابع RDO على هيئة ٣٢ بيت وتأتى فقط مع إصدارة Enterprise Edition من فيجوال بيسك.
- RDO ٢,٥ هي النسخة الحالية مع تحديث طفيف لفيجوال بيسك ٦.
- تحتوى RDO على وحدة تسمح باستخدام أفضل لوصلات ODBC بالمقارنة بـDAO/JET.
- يؤدى استخدام RDO إلى إنتاج تعليمات برمجية للمزود اقل وزنا لأنها لا تحتوى على محرك JET الموجود في DAO.
- توفر RDO الاتصال بقواعد بيانات SOL البعيدة بنفس سرعة اتصالات ODBC API المكتوبة جيدا.
- تدعم RDO مجموعة معاملات Parameters لتشغيل استعلامات RDO Query ذات معاملات تسمح لك المعاملات أيضا بإرجاع المخرجات من الإجراءات المخزنة.
- تستطيع RDO إجراء استعلامات لا تزامنيه.
- يوجد في RDO نموذج لكل الأحداث.
- تحتوى RDO على User Connection Designer وهى إدارة رسومية تستخدم في تسهيل عملية الوصول إلى البيانات.

ملاحظة لا يجب أن تكون RDO عاملا مؤثرا على قـرارك لشـراء Enterprise Edition من فيجوال بيسك. بدلا من ذلك، يجـب أن يكـون شـراء هـذه الإصدار معتمدا على حاجتك لإنتاج تطبيقات متعددة الطبقـات ذات طبقـة منظمة أو اكثر يتم إنشاؤها كمكونات تعمل من بعيد.

تعليمات برمجية نموذجية

مع RDO، يكون نموذج الوحدة منحازا أو اكثر إلى قواعد البيانات العلاقية مثل أوراكل وخـادم مايكروسوفت SQL. وتتغير المصطلحات إلى RDO Resultset بدلا من RDO Record set وColumn بدلا من Field. كما ترى في التعليمات البرمجية ١٢-٣، تقوم بإعداد وحـدة اتصـال RDO Connecting لتنفيذ الاتصال الكامل مع قاعدة بيانات بدلا مـن وحـدة قـاعدة بيانـات DAO/JET التي تنشئ أكثر من اتصال قاعدة بيانات واحدة. الشي الأخر الجدير بالملاحظة هـو أنه لدينا تنوعا اكبر من الخيارات لإنشاء RDO Results الخاصة بنا وآلتي يمكن أن تؤدى إلـى شي من الارتباك.

تعليمات برمجية ١٢-٣

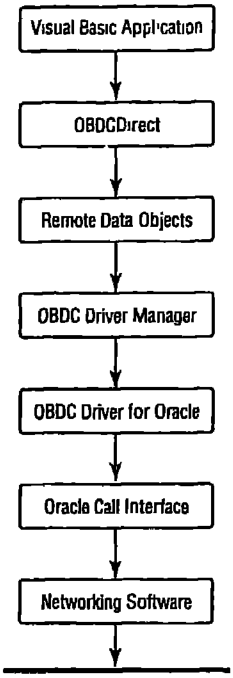
```
Private Sub FetchRDO()
   Dim rsMovies As rdoResultset
   Dim strSQL As String
   Dim iColumn As Integer
   Dim strConnection As String
   Dim cn As rdoConnection
'  Connect to the database
   With rdoEngine
      .rdoDefaultCursorDriver = rdUseClientBatch
      .rdoDefaultLoginTimeout = 1
   End With
   StrConnection = "DSN=movies;UID=video_user;PWD=ummagumma"
   Set cn = rdoEngine.rdoEnvironments(0).OpenConnection _
      ("", rdDriverNoPrompt, False, strConnection)
'  Run the query
   strSQL = "SELECT * FROM video_store.movies ORDER BY movie_id"
```

```
    Set rsMovies = cn.OpenResultset(strSQL, rdOpenForwardOnly, _
            RdConcurReDAOnly)
  ' Display the fields
    While Not rsMovies.EOF
      For iColumn = 0 To rsMovies.rdoColumns.Count - 1
        Debug.Print rsMovies(iColumn)
      Next ' iColumn
      rsMovies.MoveNext
    Wend
  Set rsMovies = Nothing
    Set cn = Nothing
  End Sub
```

ODBCDirect

تعتبر ODBC DIRECT شيئا غريبا في هذه القائمة. وهى جزء هام من RDO لأنها تعتبر غلافا على غلاف RDO. على أي حال فنموذج وحدها هو نفسه نموذج وحدة RDO أساسا توضح تطبيقات البرامج التي تستخدم في DAO "انظر الشكل ١٢-٥" هذا الطريق الملتوي، وقد أعتبر الكثيرون أن ODBCDIRECT هو هجين غير مرغوب فيه. على أي حال، فهي جزء من DAO3.51 وهى أقوي كثيرا من DAO/JET وأداء ODBC DIRECTRTS يقارب قوة أداء RDO بالرغم من وجود طبقات إضافية من البرامج وقد كانت ذات فائدة كبرى عند استخدامها داخل تطبيقات مثل مايكروسوفت اكسل حيث لا تستطيع استخدام RDO لأسباب ترخيصيه على أي حال فإن هذا العيب سيجعلك تستفيد من وحدات بيانات ActiveX Data Objects (DAO).

الشكل ١٢-٥
طبقات البرمجيات المستخدمة في ODBCDirect.



خصائص

- تتاح ODBCDirect فقط في إصدار ٣,١ من DAO ٣٢ بيت وما بعدها.
- تستخدم ODBCDirect نموذج DAO ولذلك يسهل تعليمها على المطورين الذين يعملون بالفعل مع مايكروسوفت أكسس.
- تحتوي ODBCDirect على وحدة اتصال Connection، لذلك لا تتعرض لاستخدام DAO/JET لاتصالات قاعدة بيانات متعددة.
- مساحة الذاكرة اكبر من RDO ولكنها اصغر من DAO/JET.
- تدعم ODBCDirect استعلامات DEF Query وهي مرادفة لاستعلامات RDO Query.
- تدعم ODBCDirect بعضا من مجموعة خصائص RDO بما فيها معاملات الإدخال من إجراءات مخزنة.
- تدعم ODBCDirect العمل اللا تزامني.
- لا تعرض ODBCDirect خطوات RDO.
- ODBCDirect هي طريقة نموذجية لكتابة تعليمات برمجية تنتقل بين قاعدتين بيانات أكسس وأوراكل.

تعليمات برمجية نموذجية

كما ترى في التعليمات البرمجية ١٢-٤، فإن نموذج الوحدة الذي تستخدمه ODBCDirect هو نفسه نموذج وحدة DAO/JET فيما عدا نقطيين: تدعم ODBCDirect وحدة Connection، كما أن RDO تعتمد على عدد خيارات اكبر لإنشاء مجموعات نتائج وتعتبر وحدة Connection تطورا كبيرا بالمقارنة بمطالبة DAO لاتصالات متعددة. تقوم التعليمات أولا بإنشاء وحدة Connection في مساحة عمل ODBC ثم تقوم بإنشاء مجموعة سجلات مقابلة للاتصال.

تعليمات برمجية ١٢-٤

```
Private Sub FetchODBCDirect()
   Dim cn As Connection
   Dim rsMovies As Recordset
   Dim strSQL As String
   Dim strConn As String
   Dim iField As Integer
'  Connect to the database
   DBEngine.DefaultType = dbUseODBC
   StrConn =
"ODBC;DSN=movies;UID=video_user;PWD=ummagumma;"

   Workspaces(0).DefaultCursorDriver = dbUseNoCursor
   Set cn = Workspaces(0).OpenConnection("movies",
dbDriverNoPrompt, _
          False, strConn)
'  Run the query
   strSQL = "SELECT * FROM video_store.movies ORDER BY movie_id"
   Set rsMovies = _
      cn.OpenRecordset(strSQL, dbOpenForwardOnly, 0, dbReadOnly)

'  Display the fields
   While Not rsMovies.EOF
      For iField = 0 To rsMovies.Fields.Count - 1
         Debug.Print rsMovies(iField)
```

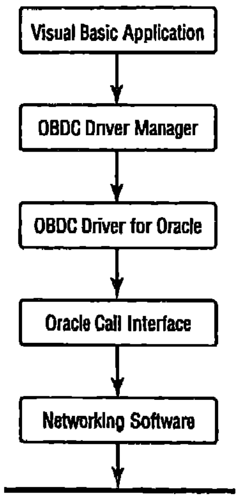
```
        Next ' iField
        rsMovies.MoveNext
    Wend

    Set rsMovies = Nothing
    Set cn = Nothing

End Sub
```

تعليمات برمجية للاتصال ODBC API

يتم تنفيذ ODBC في ويندوز كما يتم باقي العناصر، بواجهة تطبيقاتها لبرمجة التطبيق (API) توفر ODBC API طريقة أبسط لخفض استدعاءات الوظائف بواسطة واجهة تطبيق لغة C (انظر الشكل ١٢-٦). تتجاوز API الحدود المتعلقة بواجهة تطبيق المستوى الأعلى التي عادة ما يتم تقديمها (VB ويجب تبعا لذلك أن توفر مستوى أداء أعلى كثيرا. وهذه الطريقة معتادة لدى مبرمجي ويندوز الذين يستطيعون الاستفادة القصوى من API ليحصلوا على الوظائف التي ستظل غير متاحة لمبرمجين آخرين. في الواقع، لم تكن هناك بدائل كثيرة قبل فيجوال بيسك.



الشكل ١٢-٦

طبقات البرامج المستخدمة في ODBC API.

خصائص

- توجد ODBC API كاستدعاءات ١٦ بيت أو ٣٢ بيت.

- تدعم ODBC API أي وظيفة تدعمها ODBC والتي تتضمن أيا من الخصائص التي تمكنك استخدامها في طرق الوصول الأخرى:
  - استرجاع مجموعات النتائج.
  - استدعاء الإجراءات المخزنة.
  - استرجاع معاملات الإخراج من الإجراءات المخزنة.
  - التشغيل اللا تزامني.
  - التعامل مع الأحداث للوصول إلى قواعد البيانات.

العيوب

لن يقوم هذا الكتاب بتغطية برامجه ODBC API بأي تفصيل. بالرغم من وجود الكثير من المطورين الذين يستخدمون ODBC API للاتصال بقواعد بيانات بعيدة (لأنها معروفة بالسرعة والتفوق في خصائصها) فإن هناك أسبابا أخري لفعل ذلك خاصة أن لديك الآن وحدات بيانات ActiveX للأداء ووحدات أوراكل لـ OLE لدعم جميع الخصائص التي يمكن أن يقدمها خادم أوراكل. هناك عيوب ذات تأثير كبير لاستخدام ODBC API: وهى المدرجة بأسفل.

- تتطلب ODBC API تعليمات برمجية أكثر جدا من أي طريقة أخرى متاحة مع نقاط الصيانة ذات الصلة.
- تتطلب التعليمات البرمجية وقتا أطول للكتابة والاختبار.
- التعامل مع الأخطاء يتم تقييده بإدارة in-line retum-code.
- يلم القليل من المطورين إلماما تاما بـ API وهو ما يسبب مشاكل طويلة الأمد تتعلق بالصيانة.
- لم تعد مميزات أداء ODBC API كافية لتجعل منها مشروع تطوير يعتمد عليه خاصة عند مقارنتها لمجموعة سجلات للقراءة فقط وللإرسال فقط في DAO.
- مستقبل ODBC API البعيد أصبح الآن محل شك مع تقديم OLE-DB.

تعليمات برمجية نموذجية

لترى مثالا لنوع البرنامج الذي سيكون عليك كتابته بواسطة ODBC API، راجع التعليمات البرمجية ١٢-٥ توضح هذه التعليمات المجهود الكبير الذي ستبذله باستدعائك لوظائف API عند تنفيذ أي شيء. نحن ننشئ أولا مقبض بعد ذلك بإنشاء مقبض جملة SQL وتنفيذ الجملة. ثم نقوم باسترجاع الحقول في مجموعة النتائج ثم نقوم أخيرا بإغلاق الاتصال وجميع المقابض التي استخدمناها.

تعليمات برمجية ١٢-٥

```
Private Sub FetchOdbcApi()
   Dim henv As Long
   Dim hdbc As Long
   Dim hstmt As Long
   Dim rc As Integer
   Dim strConnect As String
   Dim connectOut As String * 255
   Dim connectOutLen As Integer
   Dim strSQL As String
   Dim rCols As Integer
   Dim sValue As String * 101
   Dim iCol As Integer
   Dim lValueLen As Long
   Dim ivalueLen As Integer
   Dim strfield
   rc = SQLAllocEnv(henv)
   rc = SQLAllocConnect(ByVal henv, hdbc)

   connectOutLen = 0
   connectOut = String(255, 0)
   connect = "dsn=movies;uid=video_user;pwd=ummagumma"
   rc = SQLDriverConnect(hdbc, Me.hWnd, connect, Len(strConnect), _
      connectOut, 255, connectOutLen, SQL_DRIVER_COMPLETE)
   rc = SQLAllocStmt(hdbc, hstmt)

   strSQL = "SELECT * FROM video_store.movies "
   rc = SQLExecDirect(hstmt, strSQL, Len(strSQL))
   rc = SQLNumResultCols(hstmt, rCols)
   Do
      rc = SQLFetch(hstmt)
      If rc = SQL_SUCCESS Or rc = SQL_SUCCESS_WITH_INFO Then
         Svalue = String(101, 0)
```

```
        For iCol = 1 To rCols
            Rc = SQLGetData(hstmt, i, SQL_C_CHAR, sValue, 100, _
                LvalueLen)
            If lValueLen > 0 Then
                Strfield = Left(sValue, lValueLen)
            Else
                Strfield = ""
            End If
            Debug.Print strfield
        Next
    End If

    Loop While rc = SQL_SUCCESS
    rc = SQLFreeStmt(hstmt, SQL_CLOSE)
    rc = SQLDisconnect(hdbc)
    rc = SQLFreeConnect(hdbc)
    rc = SQLFreeEnv(henv)
End Sub
```

ملاحظة مثال التعليمات في التعليمات البرمجية ١٢-٥ تم تبسيطه للتوضيح. وقد تـم بالتحديد إزالة تصحيح الأخطاء.

كما ترى، فإن استدعاء API ليس بالأمر اليسير تماما عليك. إذا كنت معتادا على البرمجة بـ DAO، فربما تكون مندهشا لمعرفتك حجم ما تقوم به ليظهر الشكل النهائي الذي تراه. عليـك أن تفهم تماما أي وظائف منخفضة المستوى يتم استخدامها لتسمح بجميع النتائج. يمكن أن يسـتغرق التعليم هنا وقتا أطول بكثير من أي طرق وصول أخري. وعلى الرغـم مـن أن الكثـير مـن المبرمجين قد أصبحوا خبراء في هذه الأساليب، فإن مميزات الكتابة API التي كانت موجودة من قبل تقل بمرور الوقت.

والسبب في تغيير أهمية API هو أنه مع تقديم RDO فإن لدينا طبقة دقيقة مغلقة علـى API تخفى جميع التعقيدات. ومميزات RDO على API هي أن الأداء تقريبا يشبه قوة الأداء مـع API وتمت كتابة طرق وصول إلى البيانات بواسطة فرق تطوير تفهم تماما كيفية تحسين الوصول إلى البيانات ويمكن أن تكون التعليمات البرمجية التي أنتجوهـا فـي نفـس سـرعة تعليمـات API،

النموذجية التي يمكن أن نكتبها أو أسرع. ومع DAO، لدينا الآن إمكانية واجهة التطبيق عالية المستوى وهى التي يمكن أن تكون أسرع من ODBC API كثيرا من الظروف الناسبة (مؤشرات للقراءة فقط وللإرسال فقط). كنتيجة لذلك، فلن يناقش هذا الكتاب ODBC API بالتفصيل. إذا كنت مصرا على استخدامها، فهناك مصادر متعددة متاحة أمامك تتضمن برنامج مطوري ODBC API من مايكروسوفت.

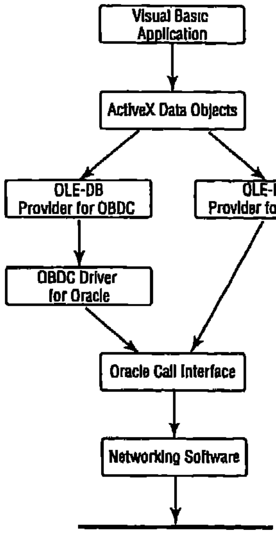
ملاحظة

مع الإمكانية الحالية لاستخدام DAO و RDO واتجاه مايكروسوفت نحو الطريقة الجديدة OLE-DB. فلا يمكن تصور بدء مشروع جديد باستخدام التعليمات البرمجية (ODBC/API) قد يشجع وجود مجموعة من خبراء برمجة API على استخدامها لكن هناك مشاكل كبرى تتعلق بالصيانة طويلة الأمد ستظل تحدث. قد يعترض بعض مبرمجي API المتمرسين، لكن تكلفه المشروع فيما يتعلق بوقت التنفيذ والصيانة واستمرارية كفاءته ستكون أكبر من أن يقبلها أي مدير مشروع.

وحدات بيانات ActiveX

وحدات بيانات "ActiveX DAO" هي نموذج مايكروسوفت الأخير والحالي للوصول إلى البيانات، لذلك لن تكون مصطلحاتها مألوفة لديك كالآخرين. كما رأينا في فصل ١١، تقنية جانب المزود، فإن ADO هي مستهلك بيانات لمزود OLE-DB يعني استخدام OLE-DB أن لدينا مصفوفة ضخمة من مصادر بيانات محتملة بحيث لا نكون محدودين بمصادر البيانات التقليدية مثل قواعد البيانات العلائقية لكننا الآن نستطيع الوصول لكل شئ من ملفات النصوص ونظم البريد الإلكتروني إلى نظام مراقبة محطة طاقة نووية.

وكمستخدم أوراكل، فأنت مهتم أساسا بهذا النوع من مصادر البيانات تستطيع الوصول إلى أوراكل من خلال مزود ODBC OLE-DB وأيضا من خلال مزود أساسي تم إصداره مع فيجوال بيسك ٦. تستطيع رؤية هذه الطرق البديلة في طبقات البرمجيات الموضحة في الشكل ١٢- ٧.



الشكل ١٢-٧
طبقات البرامج المستخدمة في DAO.

كانت ADO موجودة لبعض الوقت قبل ذلك. فقد تم إصدارها أساسا كجزء من مكونات مايكروسوفت للوصول للبيانات "MDAC" باسم ADO الذي اصبح الأسلوب الأساسي للوصول للبيانات المستخدمة مع"ASP" Active Server Pages هي لغة كتابة على الخادم تزود خادم معلومات الإنترنت لمايكروسوفت "IIS" بالقدرة على إنتاج صفحات ويب نشطة بكفاءة اكبر كثيرا من استخدام لغة كتابة CGI وهي الطريقة التقليدية. داخل نطاق عمل ASP تقوم ADO بتوفير طريقة كافية وقابلة للتدرج للوصول إلى مصادر البيانات من داخل صفحة ويب تناسب ADO أيضا لمكونات التطبيقات التي تعمل تحت خادم تحرك مايكروسوفت Microsoft Transaction Server "MTS" سنقوم بمناقشة ADO بالتفصيل في فصل ١٧، وحدات بيانات Active X.

خصائص

- تتاح ADO فقط في صورة ٣٢ بيت.
- الإصدار الحالية هي ADO ٢,٠ وهي تصدر مع فيجوال بيسك ٦.
- يؤدى استخدام ADO لتقليل وزن تعليمات المزود البرمجية وهى تناسب تطوير الويب التي تكون فيها هي النموذج لصفحات الخادم النشط "ASP" بالفعل.
- تحتوى ADO على وحدة اتصال Connection وتسمح لك بإنشاء اتصال من خلال وحدة مجموعة سجلات Recordset أو أمر Command.
- اتصال ADO بأوراكل يكون سريعا خاصة مع مزود OLE-DB الأساسي يمكن لمؤشر للقراءة فقط للإرسال فقط أن يكون أسرع من RDO بمقدار الضعف.

- تعطى ADO للمطور إمكانية الوصول لخصائص متقدمة كثيرة في قاعدة البيانات مثل "استرجاع جداول SQL/PL من خلال مشغل ODBC الصحيح".
- تدعم ADO وحدة أمر COMMAND لإعداد وحدة لاسترجاع مجموعات نتائج واستدعاء إجراءات مخزنة.
- تدعم ADO مجموعة معاملات PARAMETERS لتشغيل استعلامات COMMAND ذات معاملات.
- تستطيع ADO تنفيذ استعلامات لتزامنيه.
- يوجد نموذج أحداث كامل ADO.
- DATA ENVIRONMENT هي أداة رسومية تعتمد على ADO تستخدم لتسيير عملية الوصول إلى البيانات.

تعليمات برمجية نموذجية

توضح التعليمات البرمجية ١٢-٦ جزءا نموذجيا من تعليمات الوصول لمجموعة سجلات من ADO في هذه الحالة، يتم استخدام مزود OLE-DB الأساسي لأوراكل ويتم إعداد مجموعة السجلات افتراضياً للتقديم فقط وللقراءة فقط.

تقوم التعليمات البرمجية بإنشاء اتصال بقاعدة بيانات أوراكل ثم بإنشاء مجموعة سجلات بعد العمل داخل مجموعة السجلات والاتصال.

تعليمات برمجة ١٢-٦

```
Private Sub FetchDAO()
    Dim cn As Connection
    Dim rsMovie0s As DAODB.Recordset
    Dim strConn As String
    Dim iField As Integer
    Connect to the database
    Set cn = New DAODB.Connection
    With cn
        .CursorLocation = adUseClient
        .Provider = "MSDAORA"
    End With
```

```
    cn.Open "source=movies;user
id=video_user;password=ummagumma;"
'   Run the query
    Set rsMovies = New DAODB.Recordset
    rsMovies.Open "SELECT * FROM video_store.movies", cn

'   Display the fields
    While Not rsMovies.EOF
       For iField = 0 To rsMovies.Fields.Count - 1
          Debug.Print rsMovies(iField)
       Next ' iField
       rsMovies.MoveNext
    Wend

    Set rsMovies = Nothing
    Set cn = Nothing

End Sub
```

وحدات OLE الخاصة بأوراكل

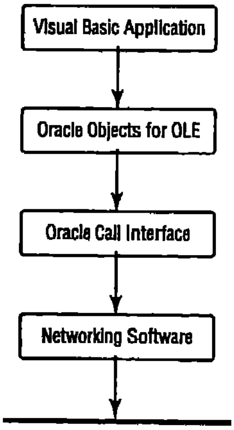
على الرغم من أن مايكروسوفت قد زودتنا بالكثير من نماذج البيانات التي تختلط علينا أسماؤها (لا تشغل نفسك بمقارنة أيها هو الأفضل لتستخدمه مع كل حالة)، فقد زودتنا شركة أوراكل بنموذج واحد أساسي هو وحدات أوراكل (OO4O) OLE وهو خادم OLE في طور التشغيل (يأتي في صورة DLL). يزود تسلسلا للوحدات مشابها تماما لـ ADO لكنه لا يستخدم نموذج ODBC فهو يتعامل مع أوراكل من خلال واجهة تطبيق استدعاء أوراكل وهو مشغل ذو تعليمات برمجية أساسية (راجع طبقات البرمجيات في الشكل ١٢- ٨) من سوء الحظ أن هذا المنتج غير واسع الاستخدام، فلن تستطيع الحصول على فاعلية اكثر شمولا مع أي منتج آخر عند الوصول لقواعد بيانات أوراكل، كما أن هذا المنتج كامل، ومستقر. لن يكون OO40 مفيدا إذا كان عليك الوصول إلى قواعد بيانات أخري، تذكر هذا إذا كان لديك موقع قواعد بيانات متعدد المصادر.

وبخلاف التقنيات الأخرى للوصول إلى قواعد البيانات فإن هناك إصـــدارات حديثـــة أكـــثر 0040 يجب عليك أن تستخدم الإصدارات الأحدث منها والتي تمثل غالبا طرقا لمعالجة الأخطاء الصغيرة وتحسينات للخصائص والأداء والتحسينات الأحدث في الإصدارات ٢,٢ و٢,٣ تعتمـــد على جعل المنتج أكثر توافقا مع بيئات مشابهة لبيئة الويب ويمكن اســتخدامه فــي MTS تدعــم الإصدار ٢,٣ أيضا وسيلة إدارة مجمع اتصال Connection Pool Management. تسمح لــك هذه الخاصية بإنشاء مجمع للوحدات المستخدمة كثيراً مثل وحدات ORA DATABASE وبذلـــك ففي بيئة يقم فيها المستخدمون دائما بالاتصال وإيقاف الاتصال (مثل تطبيق ويب)، يمكـــن إبقـــاء الوحدات مفتوحة نشطة ولإعادة اتصالها باستمرار.

بشكل عام فإن 0040 هي طريقة الوصول إلى البيانات التي ستجد فيها أكبر دعم لخصائص قاعدة بيانات أوراكل وهى أيضا المنتج الذي ربما سيكون الأول في دعم خصائص الوحـــدات ٨ على أي حال هناك نطاق واسع من الفاعلية سنراه في فصل ١٨ وحدات أوراكل OLE.

الشكل ١٢-٨

طبقات الـــبرامج المستخدمة فـــي 0040.



خصائص

- تستطيع تحميل وحدات أوراكل OLE من موقع أوراكل على الإنترنت أو أن تجدها علـى قرص خادم أوراكل الخاص بك.
- تتاح 0040 في إصدارات ١٦ بيت و٣٢ بيت (بالرغم من تحديـــد الطـــرق الأحـــدث بإصداره الـــ ٣٢ بيت).
- النسخة الحالية هي ٢,٣.
- أعداد 0040 أسهل مع وصول مباشر للاسم البديل لقاعدة البيانات.
- تسلسل وحدات 0040 معروف لدى مستخدمي RDO,DAO

- تدعم OO4O مجموعة معاملات PARAMETERS تعتمد على اتصالات والتي تصبح المتغيرات مرتبطة PL/SQL.
- تستطيع OO4O استدعاء إجراءات مخزنة بمعاملات وهو ما يسمح لك باستخدام معاملات إدخال وإخراج.
- تستطيع OO4O استرجاع جداول PL/SQL.
- تستطيع OO4O استرجاع مجموعات النتائج من إجراءات مخزنة (باستخدام متغيرات مؤشر PL/SQL.
- لا تدعم OO4O العمل اللا تزامني أو الأحداث اللا تزامنيه.
- يستفيد عنصر تحكم البيانات في OO4O من الاتصالات المشتركة.

ملاحظة يجب الإشارة إلى أنه مع إمكانية وصولك إلى قواعد بيانات أوراكل فقط باستخدام OO4O فمن الممكن نظريا الوصول إلى قواعد بيانات SQL الأخرى مثل الخادم SQL لمايكروسوفت من خلال أحد منتجات جيت واي أوراكل يجب استخدام هذه المجموعة فقط كحل أخير لأنك ستفقد قدرا كبيرا من الفاعلية مع قاعدة البيانات وراء جيت واي وسيقل الأداء عن المطلوب.

تعليمات برمجية نموذجية

يكون الاتصال في OO4O أسهل كثيرا (وأقل في الخصائص) تقوم بإدخال فقط اسم المستخدم الخاص بك وكلمة السر والاسم البديل لقاعدة البيانات نموذج الوحدات هنا يشبه نموذج DAO مع الفرق الواضح في أن جميع الوحدات يسبقها المقطع بـ "Ora". تنشئ التعليمات وحدة OraSession ووحدة OraDatabase مرتبطة بها. ثم تقوم بإنشاء OraDynaset لتحتفظ بمجموعات النتائج التي تم استرجاعها. أخيراء يتم إغلاق وحدات OraDynaset وoraDatabase.

تعليمات برمجة ١٢-٧

```
Private Sub FetchOO4O()
    Dim OraSession As Object
    Dim OraDatabase As Object
    Dim rsMovies As Object
    Dim strSQL As String
    Dim iField As Integer
```

```
' Connect to the database
  Set OraSession = CreateObject("OracleInProcServer.XoraSession")
  Set OraDatabase = OraSession.OpenDatabase("beq-local", _
           "video_user/ummagumma", ORADB_DEFAULT)
' Run the query
  strSQL = "SELECT * FROM video_store.movies ORDER BY movie_id"
  Set rsMovies = OraDatabase.CreateDynaset(strSQL,
ORADYN_REDAONLY)
' Display the fields
  While Not rsMovies.EOF
     For iField = 0 To rsMovies.Fields.Count – 1
        Debug.Print rsMovies(iField)
     Next ' iField
     RsMovies.MoveNext
  Wend
Set rsMovies = Nothing
  Set OraSession = Nothing
End Sub
```

مقارنة نماذج الوحدات

بالرغم من أننا سنتناول نماذج الوحدات المستخدمة في الأساليب المفتوحة للوصول للبيانات في الفصلين ١٤ و١٨ فإن من المناسب هنا مقارنة الوحدات المختلفة التي تعرضها نماذج الوحدات للمطورين يعطي الجدول ١٢-١ نظرة شاملة للوحدات الكبرى في كل نموذج من النماذج الخمسة التي ستناقشها بالتفصيل.ولأنه لا يوجد تقابل مباشر بين نماذج الوحدات، فإن هذا الجدول أساساً لمقارنة الأغراض على أي حال، تستطيع أن ترى أن نموذج DAO هو الأبسط بين النماذج ويستغني عن كثير من الأعداد التي تحكم معظم الطرق. يصبح من الواضح بشدة أيضاً أن DAO/JET بها أقل ما عليه وأن الكثير من الخصائص المميزة للطرق الأخرى غير موجودة.

الجدول ١٢-١ : مقارنة الوحدات الأساسية للنماذج المتنوعة للوصول إلى البيانات

DAO/Jet	RDO	ODBCDirect	ADO	OO4O
DBEngine	RdoEngine	DBEngine	N/A	OraClient

Workspace	RdoEnvironment	Workspace	N/A	OraSession
Database	N/A	Database	N/A	OraDataBase
N/A	RdoConnection	Connection	Connection	OraConnection
N/A	RdoQuery	QueryDef	Command	OraSQLStmt
N/A	RdoParameter	Parameter	Parameter	Parameter
Recordset	RdoResultset	Recordset	Recordset	OraDynaset
TableDef	RdoTable	TableDef	N/A	N/A
Field	RdoColumn	Field	Field	OraField
N/A	N/A	N/A	Property	N/A
Error	RdoError	Error	Error	LastServer Error

تصدير جدول أكسس لأوراكل

عندما تبدأ العمل مع أوراكل لأول مرة فرعاً ستريد نقل قاعدة بيانات موجودة إلى أوراكل. وقد مرت بي كثيراً حالات يتم فيها نقل جدول مايكروسوفت أكسس من موقف عمل مشترك "أو ربما نموذج أولي" إلى قاعدة بيانات مشتركة على الرغم من وجود معالجات كثيرة في خادم SQL لتكبير البيانات لتتعامل مع هذا الموقف فإن علينا تنفيذ أشياء أخرى عند العمل مع أوراكل.

تصدير مايكروسوفت أكسس

إن أحد احتمالات نقل قاعدة بيانات من أكسس لأوراكل هو استخدام وسيلة تصدير أكسس وتمون هذه العملية كالتالي:

١- قم بتشغيل قاعدة بيانات أكسس التي تريد التصدير منها (مثل NWIND.MDB).

٢- اختر علامة جدولة Tables وانقر الجدول الذي تريد نقله. في هذا المثال، تم اختيار جدول Orders.

٣- اختر خيار القائمة File>Save AS>Export.

٤- اختر "TO an External File or Data Base" وانقر OK (انظر الشكل ١٢-٩).

٥- لا تتجاهل اسفل مربع الحوار (الذي يهتم معظمه بالتصدير إلى قاعدة بيانات أكسس أخري) وفى مربع حوار "Save Table Orders In" اختر لـ Save AS Type "ODBC Data Base".

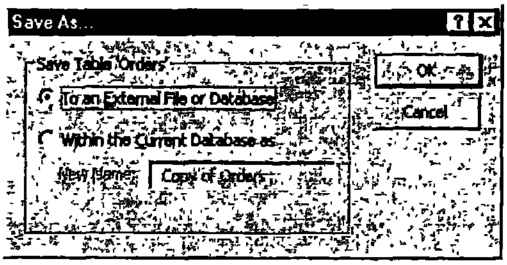
٦- حدد اسم جدول أوراكل الذي تريد التصدير إليه (انظر الشكل ١٢-١٠).

٧- اختر مصدر بيانات ODBC الخاص بك وهو قاعدة بيانات أوراكل.(في هذه الحالة، تم اختيار Movies لأنه موجود بالفعل).

٨- ادخل اسم المستخدم وكلمة السر الخاصية بك. سيتم إنشاء الجدول في مخطط قاعدة البيانات الافتراضي لاسم المستخدم الذي تختاره وسيتم استكمال عملية التصدير.

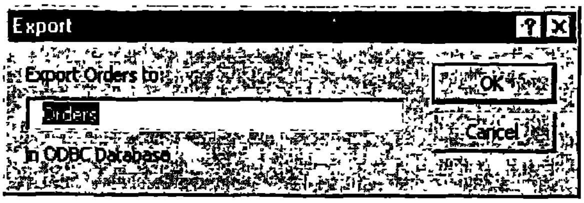
الشكل ١٢-٩

مربع حوار التصدير "Save AS"



الشكل ١٢-١٠

تحديد جدول أوراكل الذي سيستقبل تصدير أكسس



تلميح في أوراكل، يتم تخزين أسماء جميع الوحدات بحروف كبيرة. يمكن لهذا أن يسبب مشكلة إذا كان جدول الوصول أو أسماء الحقول على خليط من الحروف الصغيرة والكبيرة أو مسافات وهو ما يزيد صعوبة المشكلة. لاستخدام هذه الصيغ في أوراكل، سيكون عليك إدخال الاسم بين علامتي اقتباس مزدوجتين. هناك حل أخر أفضل هو تغيير جدول أكسس وأسماء الأعمدة لحروف كبيرة بدون وجود فارغات مضمنة قبل تصديرها، وسيكون الانتقال أسهل قليلا عندئذ.

تحذير في مثال التصدير الموضح سابقا، اختيار جدول "Orders" بالتحديد للعمل. لا تعمل بعض الجداول الأخرى لأنها تحاول إنشاء جدول بأكثر من عمود طويل جدا وهو ما يشبه أوراكل كثيرا. على أي حال، إذا كنت ستنشئ جدولا مناظرا في أوراكل، تستطيع استخدام أعمدة VARCHAR بدلا من حقول الإشعارات وسيتم عملية النقل بنجاح.

بالرغم من أننا لم نناقش استيراد قاعدة بيانات من أوراكل إلى أكسس، فإنك تستطيع تنفيذ ذلك. من المتوقع أن تكون هناك حالات قليلة تكون فيها هذه الطريقة ذات قيمة أو ربما افضل بالمقارنة بعملية عرض جدول مستورد في أوراكل.

عرض جدول مستورد في أوراكل

للتأكد من حالة الجدول المستورد، تستطيع رؤية الجداول الموجودة لديك في قاعدة بيانات أوراكل الخاصة بك بإدخال الأمر التالي في دورة SQL*Plus.

```
SELECT table_name FROM user_tables;
```

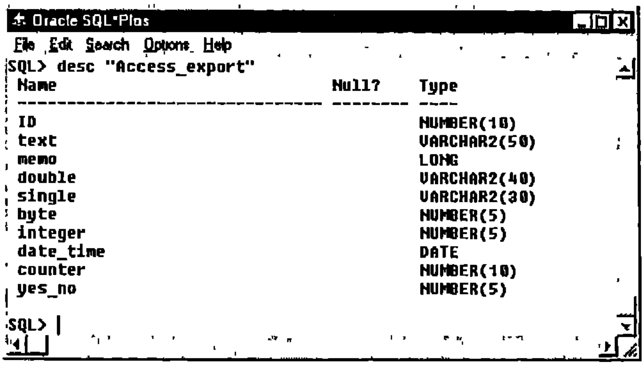
ثم لكل جدول تريده، تستطيع التأكد من صحة هيكله بعرضه بواسطة.

```
DESC table_name
```

إذا كنت قد أخطأت أو إذا كان الهيكل غير صحيح، قم بإسقاط كما يلي:

```
DROP TABLE table_name;
```

عندما تقوم بمراجعة ما سيقوم به تصدير نموذجي من مايكروسوفت أكسس، سترى أنه لا يحقق كثيراً مما ترغب في تحقيقه. في المثال التالي، قمنا بإنشاء جدول في أكسس بتسعة أنواع مختلفة من أعمدة البيانات. ثم قمنا بتصديرها إلى أوراكل لنرى الجدول الناتج الذي تم إنشاؤه. كما ترى في الشكل ١٢-١١، فإن نقل أنواع البيانات غالباً ما يكون غير مستخدم، وفى أحسن الأحوال يمكنه إنتاج تطبيق قاعدة بيانات ضعيف.



الشكل ١٢-١١
هيكل جدول مصدر من أكسس إلى أوراكل

هناك حل أفضل كثيرا قواعد بيانات أكسس إلى أوراكل وهو استخدام معالج Oracle Export Wizard For Access الذي سنعرض له في القسم التالي. اذا كنت تريد تصدير بيانات من تطبيق سطح مكتب أخر قد ترغب في كتابة تعليماتك البرمجية الخاصة بك التي تستطيع اتخاذ قرارات افضل لتحويل أنواع بيانات أكسس إلى أوراكل.على أي حال، فمن الأفضل أن تعتاد على

البرنامج مساعد SQL* Loader في فصل ٨، تحديث قاعدة البيانات، لأن هـذا البرنـامج هـو طريقة أكثر كفاءة للحصول على بيانات من تطبيق سطح مكتب إلى أوراكل.

معالج تصدير أوراكل

هناك بديل افضل لوظيفة تصدير أكسس التي رأيناها من قبل هو معـالج Migration Wizard For MS Access. يأتي هذا المنتج مع خادم أوراكل. فإذا كان لديك أوراكل سيكون لديك هـذا المعالج أيضا. يسمح لك هذا المنتج بتصدير تعريف قاعدة البيانات أو إرفاق جميع البيانات.

يتم التزويد بالمعالج كجزء من برنامج خادم أوراكل لـ NT دون مصاريف إضافية. البرامج المستخدمة لهذه الأمثلة هي لأوراكل ٧,٣. تستطيع بدء تشغيل المعـالج باختيـاره مـن خيـار Programs>Oracle Tools For NT من قائمة البداية.

١- قم بتكوين قائمة بقواعد بيانات أكسس المراد تصديرها، ثم أبدا بقاعدة بيانـات صغـيرة لأكسس حتى تكمل العملية بنجاح لمرة واحدة على الأقل (انظر الشكل ١٢-١٢).

٢- بعد ذلك، سيكون لديك فرصة لتخصيص تحويل أنواع بيانات أكسـس لأنـواع بيانـات أوراكل. ستجد أن النماذج الافتراضية جيدة بما فيه الكفاية في أغلب الأحيان.

٣- سيكون لديك الآن فرصة تخصيص أسماء وحدات البيانات. تكـون أسـماء الجـداول والأعمدة والفهارس مباشرة تماما بحيث يمكنك عادة قبول ما يقترحه المعالج. علـى أي حال، يجب عليك أن تتمهل قليلا فيما يخص Username and Tablespade Name (انظر الشكل ١٢-١٣). سيقوم المعالج بإنشاء اسم مستخدم وملف جدول وملف مضمـن جديد لك ويمكنك تغيير هذا عند هذه المرحلة.

٤- تسمح لك الصفحة التالية من المعالج بتغيير خيارات النقل إذا رغبت أنت في ذلك.

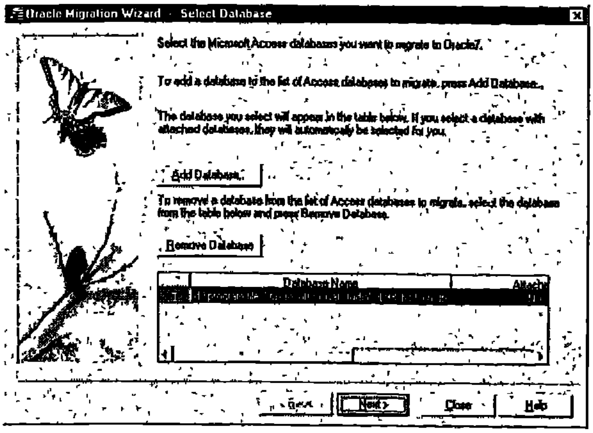
٥- يجب عليك بعد ذلك أن تقرر إذا ما كنت ستنتقل الهيكل أو الهيكل مع البيانات من قـاعدة البيانات المختارة.

٦- ادخل اسم المستخدم وكلمة السر لعملية النقل. تحتاج إلى امتيازات إدارية لتكون قـادرا على إنشاء ملفات الجداول والمستخدمين الجدد ووحدات أخري لقاعدة البيانات اسـتخدم حساب SYSTEM إذا لم يكن لديك أي أسماء مستخدمين أخري مـع إعـداد امتيـازات DBA.

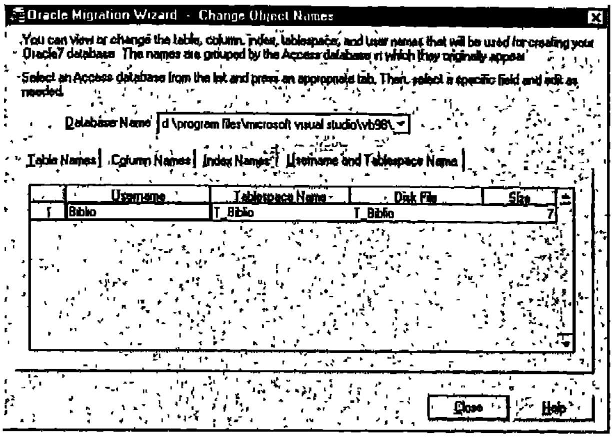
٧- عندما تنقر OK ستبدأ العملية. توقيع أن يأخذ هذا عدة دقائق لقاعدة بيانات صغيرة ووقتا أكبر إذا كنت تريد نقل قاعة بيانات اكبر.

٨- عند إتمام العملية، سيكون لديك الخيار لتراجع العديد من التقارير كأرشيف بأي أخطـاء وسكر يبت SQL لإنشاء وحدات قاعدة البيانات.

الشكل ١٢-١٢ بناء قائمة قواعد بيانات أكسس المواد تصديرها.



الشكل ١٢-١٣ تغيير الأسماء والخصائص الافتراضية المنشأة لوحدات قاعدة بيانات أوراكل.



إذا جربت ذلك ستلاحظ أن قاعدة البيانات NWIND MDB تم تصديرها بشكل صحيح. وتكون التقارير على نفس درجة تفوق الأداة. تتضمن المميزات العديدة الأخرى ما يلي:

- إدارة التكامل المرجعي.
- السماح لك باستخدام نماذج أكسس وعمل تقارير عن قاعدة بيانات أوراكل جديدة.
- العمل مع قواعد البيانات ذات الجداول المرتبطة.
- العمل مع أكسس ٢,٠ (١٦ بيت) وأكسس ٧,٠ (٣٢ بيت).

خُلاصة

قمنا بتقديم العديد من الطرق المختلفة للوصول إلي البيانات من فيجوال بيسك، و لم نذكر تقريبا طرقا أخري غير معروفة. فأي الطرق تستخدمها أنت؟ يساوى السؤال وهذا السؤال ٦٤٠٠٠ دولار، والإجابة عليه ربما تكون أكثر صعوبة وأهمية مع الوضع في الاعتبار تكاليف تطوير البرامج الآن. إذا كان لديك الوقت، حاول أن تعتاد على الكثير من هذه الأساليب بقدر الإمكان. ربما ستجد أن الأداء أو الإمكانيات لها تأثير قوى، كما ستتدخل التفصيلات الشخصية أيضا في هذا. قد يكون الاختيار شخصيا، لكن عليك ألا تبنى قرارك على الخوف أو عدم المعرفة بواحد أو أكثر من المنتجات. يقدم الجداول ١٢-٢ ملخصا شخصيا للمنتجات.

الجدول ١٢-٢: التقنيات المتاحة للوصول إلي قواعد بيانات أوراكل من فيجوال بيسك

التقنية	تقييمها
عناصر التحكم في البيانات	تعتبر طريقة سهلة للبداية لكن بها عيوبا كبيرة لا يجب استخدامها لأي شي سوى كتمرين قصير على التعليم أو على إنشاء النماذج الأولية
DAO	توفر الوصول إي أساسيات أوراكل لكنها لن تعطيك إي فائدة من خصائص أوراكل المتطورة المتاحة الآن. لا تبدأ مشروعاً جديداً بـ DAO
RDO	هي تطوير بسيط لـ DAO، وقد كانت مفضلة لدى مايكروسوفت حتى ظهور DAO أخيراً
ODBCDirect	هي أيضا تطوير بسيط لـDAO، وهى تفيد كثيراً في توفير واجهة تطبيق ODBC دون ما تتطلبه RDO. ومع وجود DAO، ربما ستخفت شهرة ODBCDIRECT
ADO	وهى الآن المنتج الأشهر من مايكروسوفت حيث وفرت الشركة قدراً كبيرا من السهولة لإنشاء قواعد بيانات أوراكل. فهي سريعة وفعالة ومناسبة لقواعد البيانات المتطورة بالإضافة إلي برمجة الويب

الجدول ١٢-٢: التقنيات المتاحة للوصول إلي قواعد بيانات أوراكل من فيجوال بيسك

التقنية	تقييمها
ODBC API	وهى أسلوب برمجة فعال لكنه معتمد. وهو يحتوى على مميزات كثيرة بفصل قدر كبير من التطور المعقد وهى مفضلة فقط إذا كنت تستخدم API بالفعل
OO40	هي المنتج الأشهر لأوراكل، وهى تدعم إمكانيات أوراكل أكثر من أي منتج أخر. وهى أيضا بيئة مستقرة وناضحة والتي سيتم تمديدها أخيرا لتغطى خصائص أوراكل المتعلقة بالوحدات

إذن، أي هذه الطرق تختار ؟ إذا كان لديك تطبيق حالي في البرامج مع خبرة المطور، فليس من اللازم تغيير الطريقة لمجرد التغيير. ستأخذك الفصول من ١٤ إلى ١٨ خلال إمكانيات التقنية الحالية التي تستخدمها وستساعدك في الحصول على الفائدة القصوى منها. ستمهد هذه الفصول الطريق أمامك لمعرفة تقنيات لم تفكر فيها أو تستخدمها من قبل.

عندما تقوم ببداية مشاريع تطوير جديدة، ينبغي لك استخدام ADO إذا كنت تريد الوصول إلي كثير من أوراكل أو إذا أردت البقاء كمنتج مايكروسوفت صف المزود إما OO40 فتستخدمه إذا أردت الحصول على أعلي كفاءة من قاعدة بيانات أوراكل الخاصة بك وكنت متا كدا من انك لن تستخدم أي قاعدة بيانات أخري. يكون OO40 مفيدا على وجه الخصوص إذا كنت تعرف أنك ستريد استخدام خصائص قواعد بيانات أوراكل الجديدة والتي ستتعرف وقتا لتصل إلي مزود OLE-DB. قم بدمج واحد من هذه التقنيات المختارة مع طريقة تعتمد على المكونات وستكون جاهزا لكل ما يستجد.

هل يجب عليك تغير التقنية التي تستخدمها؟ ربما تكون الإجابة هي نعم على الرغم من أن تحديد وقت هذا التغيير هو القرار الأصعب. يجب أن تكون على علم بخصائص كل تقنية وان تدرك أن الانتقال بين أي من هذه التقنيات ليس بالأمر ليسير (حتى بين DAO/JET وإذا كنت تريد أن تجعل الانتقال ذا مغزى). قم بتخطيط التطوير الخاص بك ألان ODBCDirect مع تذكر انك ستقوم بالتغيير في النهاية، حيث يجب أن تقلل الطريقة المعتمدة على الوحدة من أي اضطرابات لأدني حد.

١٣

موضوعات عن التصميم باستخدام فيجوال بيسك

بما أنه قد تم إيضاح تقنيات الدخول إلى البيانات في قاعدة بيانات أوراكل من داخل تطبيقات الفيجوال بيسك، فإنه من الضروري إلقاء نظرة خاطفة على أنواع الوحدات التي قد يحتاجها المستخدم للتعامل مع أي طريقة يتم اختيارها ومن الضروري معرفة بعض الموضوعات الأساسية الخاصة بالتصميم والتي يتم إيضاحها في الأجزاء التالية. وتتضمن هذه الموضوعات:

- تسجيل الدخول إلى قاعدة البيانات.
- الجداول الافتراضية والمرادفات ومخطط قاعدة البيانات.
- اختيار المؤشر.
- الاستعلام عن قاعدة البيانات.
- تحديث قاعدة البيانات.
- التحكم في عدد المعاملات.
- موضوعات التأمين.
- استخدام بيانات BLOB-type.
- معالجة الأخطاء.

القضية البنيوية

لا يتناول هذا الكتاب الموضوعات البنيوية. لذلك يجب أن تدور فكرة التصميم في المستقبل حول المكونات. يمكن أن يتم تمثيل التصميم في الفيجوال بيسك ببساطه عن طريق فصول فردية تغلف الدخول إلى أنواع معينة من البيانات. فعلى سبيل المثال من الممكن أن يتعامل فصل واحد مع حفظ البيانات على جدول MOVIES بالإضافة إلى إمكانية استعادة البيانات منه مرة أخرى ولا يجب على المستخدم فصل هذا الفصل عن مكتبة الربط الحيوي DLL، أي أنه يمكن تشغيلها من داخل برنامج المزود الأساسي وسوف تظل تصميماً يخضع لطبقات وبما أن المستخدم لديه التصميم الصحيح، فإنه من السهل رفعة بسهولة ووضع المكونات في جهاز آخر في وقت لاحق وسوف يجد المستخدم أن تطبيق أوراكل سيحتاج إلى الناحية البنيوية إن آجلاً أو عاجلاً، لذلك وضع هذا في الاعتبار عند التصميم من البداية.

تسجيل الدخول إلى قاعدة البيانات

عند الدخول إلى قاعدة بيانات لأوراكل فإن التطبيق يجب أن يمر بعملية تسجيل الدخول ولا تعني هذه العملية أنه من الضروري أن يرى المستخدم شاشة تسجيل الدخول ولكن من الممكن ذكر اسم المستخدم وكلمة المرور في التعليمات البرمجية الخاصة بالتطبيق. ويتضح أن هذا ليس هو أكثر المواقف أمناً ولكن إذا ما ترائي للمستخدم أن هذه هي أفضل الطرق لتشغيل التعليمات البرمجية فإنه يجب عليه أن يستخدم أمن نظام التشغيل للتأكد من أن التعليمات البرمجية الخاصة بالتطبيق -ومن ثم كلمة المرور- ليست متاحة لأحد.

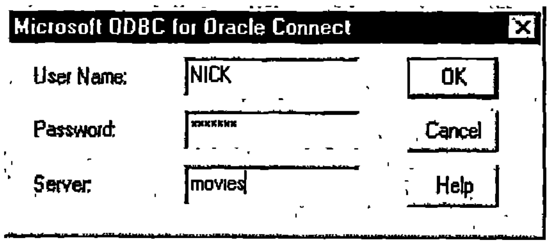
ملاحظة

قد يحتاج المستخدم إلى تضمين مستخدم عام وكلمة مرور في التعليمات البرمجية الخاصة بالتطبيق عند كتابة مكونات دخول للبيانات والتي تعمل في الطبقات الوسطى من تطبيق متعدد الطبقات ويمكن أن تتم مشاركة هذه المكونات بين عدة مستخدمين ولكن يجب التأكد من حماية التعليمات البرمجية.

وعلى الرغم من ذلك، فإن المستخدم سوف يجد نافذة تسجيل الدخول في مدير مشغل ODBC. وقد يختلف تصميم النافذة من موزع لآخر وتظهر نافذة Microsoft ODBC Driver For Oracle Connect في الشكل ١٣-١.

الشكل ١٣-١

نافذة تسجيل الدخول إلى أوراكل.



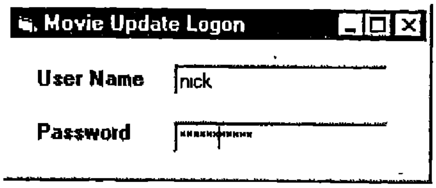
لبدء تسجيل الدخول إلى أوراكل يجب إدخال اسم المستخدم وكلمة المرور المعطاة إليك من DBA وفي بعض الأحيان ستتاح لك الفرصة لإدخال سلسلة اتصال مختلفة عن التي تم إعدادها باستخدام اتصال ODBC أو بشكل أكثر تعقيد من الذي قد تم تعريفه في التعليمات البرمجية ومن المفضل تجنب مثل هذه المواقف لأنها سوف تؤدي إلى ارتباك المستخدم في أغلب الأحوال.

ومن الممكن تجنب ظهور نافذة تسجيل الدخول عن طريق إنشاء نافذة الدخول الخاصة بالتعليمات البرمجية الموضحة في الشكل ١٣-٢ ومن الممكن تغيير المربع النصي بسهولة إلى مربع لكلمة المرور عن طريق تحديد ملكية كلمة المرور وتكون "*" وهناك بعض الفرص لذلك:

- من الممكن إدخال اسم المستخدم إلى الطبقة الوسيطة لتسجيل الدخول الفعلي أما إذا كانت هناك رغبة في جعل الطبقة الوسيطة تسجل الدخول للمستخدم، فيجب أن يكون هناك اسم مستخدم محدد وكلمة مرور محددة أو تمرير المعلومات إلى المكونات. ولا يمكن للمستخدم أن يستخدم حاث الطبقة الوسيطة مباشرةً لأية معلومات عن أي مستخدم.
- تتيح شاشة تسجيل الدخول الخاصة بالمستخدم للمستخدم إعاقة كلمة المرور حيث أنه لا يمكن إيجاد أية طريقة أخرى في أوراكل وتتيح هذه الطريقة كتابة التعليمات البرمجية الخاصة بالمستخدمين لتحديث كلمات المرور الخاصة بهم على سبيل المثال وبذلك يمكن التأكد من أن كلمة المرور الجديدة تختلف عن كلمة المرور الموجودة بالفعل.

الشكل ١٣-٢

نافذة تسجيل الدخول لتطبيق MOVIES.

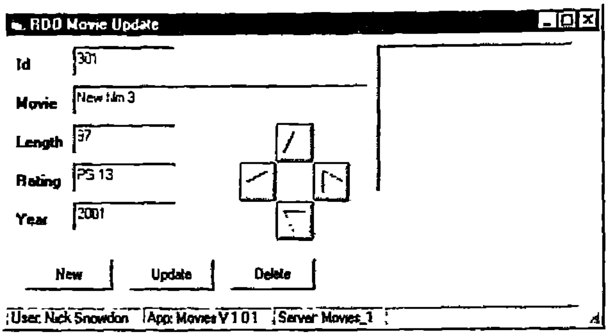


ملاحظة

السبب وراء إعاقة اسم المستخدم وكلمة المرور كان عندما تحتم على أحد التطبيقات إرسال بيانات من قاعدة البيانات إلي خادم آخر عبر FTP فرغم أن ذلك قد تم بأمان، إلا أنه كان لابد من إيجاد تسجيل دخول أخر لخادمFTP ولتجنب غضب المستخدمين، فقد تم التأكد من أن الخادمين قد سمحا بنفس كلمتي المرور، لذلك فإنه عند استخدام القيمة CACHED لم يكن هناك حاجة لتسجيل الدخول يدويا للمرة الثانية.

ويعد تضمين شريط الحالة المزود بالمعلومات الهامة من أحد الطرق لتحسين الفائدة العامة من النافذة الأساسية للتطبيق وعلى الأقل، إذا كانت هناك رغبة لتضمين اسم أحد المستخدمين، عند تسجيل الدخول، وفي إحدى البيئات التي تسمح بالدخول إلى أكثر من نموذج لأوراكل، فإن يتم تضمين اسم خدمة النموذج. فضلا عن أن إضافة إصدار التطبيق الذي تم تشغيله يعد لمسة جمالية. وعند استخدام الحد الأدنى لرسائل الأخطاء "راجع قسم "معالجة الأخطاء" في هذا الفصل" فإنه يمكن وضع رسائل الأخطاء أيضا في شريط الحالة، ومن الممكن وضعها بلون أحمر حتى

تكون واضحة. وبذلك تصبح +نافذة التطبيق الأساسية قريبه من الشكل ١٣-٣ ويتـــم اســتخدام التعليمات البرمجية الموضحة في الجدول ١٣-١. حتى يتسنى الحصول على قيم شريط الحالة.



الشكل ١٣-٣

نافذة التطبيق الأصلية متضمنة شريط الحالة.

الجدول ١٣-١: العثور على معلومات التطبيق وقاعدة البيانات لشريط الحالة.

المعلومة	كيفية الحصول عليها
رقم إصدار التطبيق	APP.MAJOR & ". " & APP.MINOR & "." & APP REVISION
اسم المستخدم	يتم الاختيار من DUAL
اسم قاعدة البيانات	يتم الاختيار من GLOBA1-NAME

جداول اعتباريه ومرادفات ومخطط قاعدة البيانات

يعد الدخول إلى الجداول الاعتبارية والمرادفات ومخطط قاعدة البيانات من أحد المشـــاكل التـــي واجهت طرق مايكروسوفت لإدخال البيانات وقد انحصرت هذه المشاكل حول صفـــات مشـــغلي ODBC المتاحة، ويجب الإشارة إلى أن وحدات أوراكل لوحدات OLE والتي سوف يتم التعـوض لها في الفصل ١٨ لا تحتوي على مثل هذه المشاكل ويبدو هذا طبيعيا لأنـــها طريقـــة أوراكـــل الخاصة للدخول.

غير أن هذه المشكلة تظل بغير حل لأن الكثير من التركيبات تستخدم الجداول الاعتباريـــة أو المرادفات لإخفاء الجداول الأساسية للاستعلامات وفي حالة عدم استخدام المرادفات يجب علـــى المستخدم أن يصبح قادراً على تعريف مخطط قاعدة البيانات ،أن التعليمات البرمجية تشير إلـــى وحدات مخطط قاعدة البيانات المراد الدخول إليها: "مثل الجداول أو إجراءات التخزين".

وقد استطاع مشغل ODBC للأوراكل والذي يصاحب الإصدار السادس من برنامج الفيجوال بيسك وأحدث الطرق للدخول إلى قواعد البيانات حل معظم هذه المشاكل وأصبح الآن بالإمكان استخدام الجداول الفرعية والمرادفات ومخطط قاعدة البيانات في كل أوجه الدخول إلى ODBC والممول ODBC الأصلي الخاص بأوراكل والذي يصاحب الإصدار السادس من برنامج فيجوال بيسك لا يحوي أيضاً أية مشاكل.

ولكن على الرغم من ذلك تظهر مشكلة عند محاولة الدخول إلى حزم PL/SQL من خلال ODBC وإذا كانت هناك حاجة في الوقت الحالي إلى بعض الحزم فإن المستخدم قد يحتاج إلى مشغل أقدم من ODBC أو قد يحتاج المستخدم أن يتحول إلى ADO أو OO4O ويجب ملاحظة أنه عند تحديد أحد الإجراءات المخزنة في إحدى الحزم فإنه يجب استخدام الصيغة *schema.package procedure* بدلاً من الصيغة العادية *schema.procedure*.

ملاحظة تتضح هنا إحدى خصائص DAO/JET، حيث يجب وضع اسم كل من SCHEMA والجدول في أقواس مربعة مثل SELECT * FROM [00video_store. Movies] عند استخدام أسماء SCHEMA في الاستعلام بدون SQLPassThrough.

اختيار المؤشر

يجد القارئ في الجزء الثاني من هذا الكتاب تركيزا خاصا على المؤشر ويوفر أوراكل مجموعات للنتائج للأمام فقط وأيضا للقراءة فقط عند تنفيذ أحد الاستعلامات وأي مؤشر قابل للإزاحة يتم تنفيذه بواسطة المزود وإذا كانت هناك رغبة لاستخدام هذه الميزات الإضافية فإن هذا التطبيق من الممكن أن يعرض معلومات غير حديثة كمل انه سوف يكون أكثر عرضة لتحديث الاضطرابات كنتيجة لذلك فإن هذا التطبيق لن يصبح قابلا للقياس فيجب على المستخدم أن يقلل من التعامل مع قاعدة البيانات قدر الإمكان كما ينصح بالدخول والخروج بأسرع ما يمكن من خادم أوراكل. ويجب أن تتم التعاملات بشكل سريع لتقليل مواقف القفل. ومن ثم فنحن نوفر مقدمات للمؤشر النهائي.

المؤشر النهائي

تصميم المؤشر النهائي للأداء والقياس ليس مؤشراً بالشكل المتعارف عليه. فيجب اختيار مؤشر للأمام فقط وللقراءة فقط. أما إذا كانت هناك رغبه في استخدام ميزة الإزاحة، فيمكن وضع كل المعلومات تحت التحكم مثل إحدى الشبكات أو الجداول الإلكترونية. وعند الاحتياج للتحديث فإنه يتم استخدام أحد الإجراءات المخزنة أو أحد نماذج مكونات الطبقة الوسطى كما يمكن تحديث الشبكة عند الرغبة باستخدام الأمر UPDATE. وسوف تتيح هذه التقنية نسخ مجموعة من النتائج القابلة للتحديث بشكل جيد. وإذا تم إعادة الاستعلام عن مجموعة التحديث بدلاً من تحديثها ،فإن ذلك يعني القرب من نسخ مؤشر فعال ويجب ألا يكون الفائض عالياً جداً ما دامت مجموعة النتائج صغيرة.

وجب أن يكون التواصل بين المكونات كافياً وبسيطاً قدر الإمكان وخاصة تحريك تمثيل الكثير من الصفوف والأعمدة وأفضل الحلول لذلك هو المصفوفات المتنوعة وهي طريقة للقراءة فقط غير أن المصفوفة تتلاءم مع التصميم النهائي. وتتميز المصفوفات المتنوعة بأنها مرنة وقابلة للتحديث لتتواءم مع أي تصميم سواء عند استخدام منهج الطبقة الوحيدة أو Microsoft Transaction Server.

الاستعلام عن قاعدة البيانات

هناك استخدام بسيط لأحد تطبيقات قاعدة البيانات التي لا يمكن أن تقرأ البيانات من قاعدة البيانات. وسوف تتم استعادة البيانات من أوراكل بشكل طبيعي باستخدام SQL QUERIES والتي تم تغطيتها تفصيليا في الفصل السابع "الاستعلامات".وبما أن الحديث سوف يتطرق إلى الفيجوال بيسك. يجب وضع استعلامات SQL داخل التعليمات البرمجية الخاصة بالفيجوال بيسك. وعادة ما تتم هذه العملية عن طريق كشوف الحساب التي تكون السجلات RECORDSETS أو rdoResultsets في RDO وoradynasets في OO4O) واستخدام السجلات Recordset هو طريقة سهلة لإنشاء إحدى الوحدات والتي تيسر الدخول إلى كل البيانات في الاستعلام.

وإنشاء سجلات Recordsets في غاية البساطة تماما مثل DAO ولكن من الممكن أيضا استعادة مجموعات النتيجة من الإجراءات المخزنة بسهوله. وهذه الخاصية، والتي يرجع تاريخها إلى الوقت الذي كان فيه Microsoft SQL Server وsybase هما نفس المنتج، مبني على أساس أن أية عبارة اختيار Select في الإجراء المخزن تعيد نتيجة هذا الاختيار Select في

السجلات Recordset افتراضيا ويتم استغلال هذه الوظيفة على نحو أكثر في RDO عن طريق السماح باستعادة أكثر من rdoRESULTSET واحد من أحد الإجراءات المخزنة في نفس الوقت. غير أنه لسوء الحظ، فإن اوراكل لا يسمح بعبارة اختيار SELECT في أحد الإجراءات المخزنة "رغم وجود الإصدار SELECT INTO المحدود كما اتضح في الفصل ١٠ "تكوين الإجراءات المخزنة" وكنتيجة لذلك لا توجد هناك طريقة سهلة لإعادة إحدى مجموعات النتائج من أحد الإجراءات باستخدام أوراكل. كما أن rdoResultset المتعدد لا يمكن الوصول إليه.

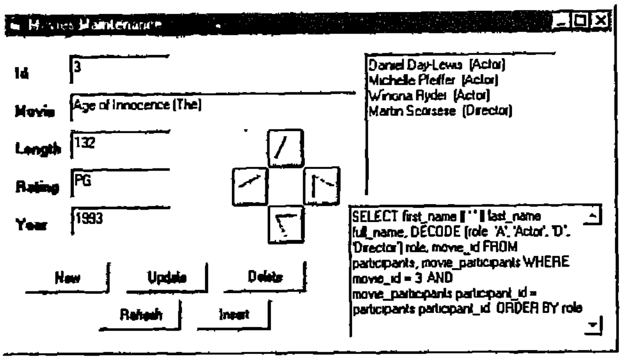
ملاحظة تتيح الإصدارات الحديثة من ADO وOO4O محاكاة مجموعات النتيجة العائدة. رغم أن هذه التقنية أكثر تعقيدا. راجع الفصل ١٧ "وحدات بيانات Active X" والفصل ١٨ "وحدات أوراكل التي تخص OLE" حيث يظهر الفرق بين المنهجين واضحا.

ويجب تذكر أنه نادرا ما يتم وضع العلامات البرمجية لسلاسل Select للاستعلامات على النحو الأكمل. وعادة ما يتم بناؤها قطعة وقد تأخذ عملية إنشاء السلاسل سطورا عديدة لذلك فمن المفضل أن يتم تقسيم السلاسل إلى عبارات منفصلة بدلا من استخدام سطور متتالية على نحو مبالغ فيه "راجع تعليمات برمجة ١٣-١". ويجب معرفة أنه عادة ما يتم إنشاء السلاسل باستخدام صياغة أوراكل. ويجب التأكد من أن القيم التي يتم الحصول عليها من المربعات النصية للفيجوال بيسك ومربعات القوائم وما شابهها يتم التعامل معها على أنها النوع الصحيح من البيانات. ويجب تضمين حرف السلسلة على وجه الخصوص في علامات اقتباس فردية كما يجب التأكد من أن علامات الاقتباس الفردية داخل إحدى السلاسل قد تم تحويلها إلى علامتي اقتباس فرديتين وأيضا التأكد من أن التواريخ قد تم تعديلها "راجع الفصل ٧ "الاستعلامات" لمزيد من المعلومات ومعرفة وظيفتين للفيجوال بيسك لتعديل التواريخ".

تعليمات برمجة ١٣-١

```
' Set up the SQL string
  strSQL = "SELECT * FROM movies "
  strSQL = strSQL & " WHERE rating = '" & cmbRating.text & "'"
  strSQL = strSQL & "  AND year > " & txtYear
  strSQL = strSQL & "  ORDER BY movie_name"
```

ويجب تذكر أن نفس السلسلة بعينها يجب أن تعمل في دورات SQL* PLUS وقد وجد أنــه من المفيد وضع مربع نصي إزاحي يحتوي على أكثر مــن سـطر علــى أيــة صيغــة تنفيـذ استعلام.ومن الطبيعي أن لا يكون هذا المربع النصي مرئياً. ويسمح للمستخدم بفتح معالج أخطـاء الإخراج والذي يعد هذا المربع النصي أحد مصادرة كجزء من تفضيلات التطبيق، وفــي هـذه الحالة سوف يصبح شكل المربع النصي مرئياً. ويجب أيضاً التأكد من إمكانية نسخ السلسلة إلــى حافظة النوافذ، ومن ثم يمكن تشغيل الاستعلام المضبوط في SQL * PLUS كما يمكــن أيضــا استخدامها لعرض إحدى خطط التنفيذ وهو ما يجعلك موضعاً للحسد من هؤلاء الذين يسـتخدمون ORACLE DEVELOPER 2000.



الشكل ١٣-٤
عــرض سلســلة SQL في صيغــة وضع Debug.

قد تظهر التعليمات البرمجية الذي تم تنفيذه مثل التعليمات برمجة ١٣-٢ "في هذه الحالة يكون RDO". يجب ملاحظة أن سلسلة SQL يتم وضعها أيضاً في الحافظة حتى يتسنى وضعـــها فـــي تطبيق آخر أو أداة تحليل.

تعليمات برمجة ١٣-٢

```
Set rsParticipants = cn.OpenResultset(strSQL, rdOpenStatic, _
            rdConcurReadOnly)

txtSQL = strSQL
If Bdebug Then
'  Display the SQL string if in debug mode
   txtSQL.Visible = True
   Clipboard.SetText strSQL   ' For the really lazy !
```

```
Else
   TxtSQL.Visible = False
End If
```

إذا كنت تعمل في بيئة متطورة يمكن كتابة سلسلة SQL ببساطة على نافذة المشاهدة باستخدام DEBUG.PRINT وبالرغم من ذلك، فإن طريقة المربع النصي تعد أكثر مرونة.

وتظهر إحدى المميزات الحقيقية في استخدام أوامر SQL في إعداد أحد العناصر التي تستطيع التعامل مع الاستعلام وتوزعها محلياً وتتيح معظم الطرق للدخول إلى البيانات عمل هذه الخطوات ويجب على المستخدم أن يستفيد من هذه الملامح على الوجه الأكمل. وهذه.الأدوات هي:

- QueryDef (ODBCDirect).
- RdoQuerry (RDO).
- Command (ADO).
- OraSQLSt (OO4O).

ويتم شرح التقنيات المستخدمة لأي أمر من أوامر SQL والتي يتم تنفيذها عدة مرات بشكل مفصل في الفصول التالية.

تحديث قاعدة البيانات

يحتاج المستخدم بعض الوقت في مرحلة تصميم التطبيق عند شروعه في تحديث قاعدة البيانات الخاصة بأوراكل لتقرير كيفية التعامل مع هذه العملية. وقد تم التعرض لهذا الموضوع ولكن من وجهة نظر أوراكل في الفصل ٨ "تحديث قاعدة البيانات" غير أن تحديد كيفية التعامل بسهولة مع الاستخدام وعنصر الأمن يعد أمراً في غاية الأهمية في هذه المرحلة.

وتتأتى سهولة تحديث إحدى قواعد البيانات بكل تأكيد من أدوات التحكم في البيانات Data Controls وتحكمات الحدود الأخرى وهي في غاية البساطة مع مجموعات النتيجة القابلة للتحديث وهناك طريقتان لحل مثل هذه المشكلة ولكن كلاهما تحتاج إلى عمل جاد من المبرمج وأول هذه الطرق لعمل هذا التحديث هو تنفيذ تعليمات INSERT,UPDATE, DELETE SQL من تطبيقات فيجوال بيسك (قد يتم هذا على مستوى المزود أو على الطبقة الوسطى من تطبيق متعدد الطبقات) غير أن عمل كل هذا من خلال الإجراءات المخزنة هو أكثر الطرق أماناً.

وإذا كان لدى المستخدم بيئة لقاعدة بيانات أحد المشروعات الموجودة وكان هناك تركيز شديد على عنصر الأمن في هذا المشروع ،فإنه من المرجح أن التحديث الوحيد المسموح به في هــذه الحالة سوف يتم عن طريق الإجراءات المخزنة. أما في حالة إعداد بيئة جديــدة وكــان هنــاك تعليمات أمنية مشددة فيما يخص تأمين البيانات، فإنه يجب على المستخدم أن يقضي بعض الوقت لتعلم كيفية كتابة الإجراءات المخزنة "راجع الفصل التاسع والعاشر" ثم حاول تطبيقــهما بدقــة. ويتضمن الفصل الرابع عشر "وحدات الدخول إلى البيانات" أحد التطبيقات القصيرة الذي من شأنه أن يسهل إجراءPL/SQL لكل عملية.

التحكم في المعاملات

تم التعرض في الفصل ٨ من هذا الكتاب "تحديث قاعدة البيانات" إلى كيفية تعامل أوراكــل مــع التحكم في المعاملات غير أن برنامج فيجوال بيسك يستطيع أيضا التعامل مع هذه العملية أيضــا عن طريق ODBC. إلا أن هاتان الطريقتان تختلفان تماما. ومن المفترض أن كل من برنــامجي ODBC وفيجوال بيسك يتم تشغيلهما في حالة Autocommit وهو الذي يعني بدوره أن أية تغيير يحدث ينتقل تلقائيا إلى قاعدة البيانات بمجرد حدوثه والسبب وراء سلوك ODBC بهذه الطريقــة هو أنها الحالة الافتراضية لقواعد بيانات Microsoft SQL Server. ومن الممكن استيعاب هــذه الخاصية خاصة عند التعامل مع مجموعات النتيجة القابلة للتحديث ويحدث التغيير في الحال عنـد استخدام طريقة Update. وليس هناك أي شيء آخر يجب عمله على المبرمج أكثر من ذلك.

ولكن يجب التأكد من وضع الحد الأدنى من عبارات DML في المعاملة الضرورية لقواعــد العمل بغض النظر عن الطريقة المستخدمة.

استخدام برنامج الفيجوال بيسك للتحكم في المعاملات:

يحتاج المستخدم إلى استخدام ثلاثة أوامر حتى يتسنى له تنفيذ عملية التحكم في المعاملة. وهــذه الأوامر الثلاثة هي نفسها الأوامر المستخدمة في طرق الدخول إلى قاعدة البيانــات التــي يتـم مناقشتها في هذا الكتاب:

BeginTrans: وهي علامة بداية المعاملة.

CommitTrans: تنفذ التغييرات على قاعدة البيانات (ينفذ أمرCOMMIT الخــاص بأوراكل).

Rollback: وهو يلغي أية تغييرات لم يتم تطبيقها (ينفذ أمر ROLLBACK الخــاص بأوراكل).

ملاحظة

تبدو طريقة BeginTrans مصطنعة إلى حد ما خاصة عند التعامل مع أوراكل، لذلك فإنه من المفضل استخدام هذه الطريقة كوسيلة لإغلاق Autocommit بدلا من اعتبارها خطوة فعالة في حد ذاتها.

ويتم استخدام هذه الأوامر الثلاثة في حالة التحديث فقط ورغم أن مجموعة النتيجة تظهر التغييرات، غير انه من غير الممكن مشاهدتها بواسطة مستخدمين آخرين إلى أن يتم تنفيذ طريقة CommiTtrans والصيغة الوحيدة المفقودة من طريقة التحكم في المعاملات هي عدم القدرة على تضمين المعاملات بنفس الطريقة التي تتم بها تلك العملية عن طريق الأمر SET TRANSACTION. الخاص بأوراكل.

وفي تعليمات البرمجة ١٣-٣ يجد القارئ أحد الأمثلة على كيفية التحكم في المعاملات من واجهة المستخدم فيما يخص DAO .ويتحكم في بداية ونهاية المعاملة زران. وفي الواقع قد يتم وضع التعليمات البرمجية داخل المعاملة داخل التطبيق في أحد الأماكن الملائمة لقواعد العمل ويجب ملاحظة أي الوحدات يتم استخدامها للتحكم في المعاملة ،في هذه الحالة تصبح هي مساحة العمل الحالية. ففيRDO سوف تكون وحداتConnection أو Environment وتكون وحدة Connection فيADO.

تعليمات برمجة ١٣-٣

```
Private Sub cmdTransaction_Click(Index As Integer)
If Index = 0 Then
Workspaces(0).BeginTrans
Else
Workspaces(0).CommitTrans
End If
End Sub
```

وإذا تم عمل عدة عمليات تحديث متكررة استيراد أحد ملفات البيانات إلى قاعدة البيانات على سبيل المثال فإنه من الممكن أن يؤدي دمج وحدة نوع استعلام مع التحكم في المعاملة. إلى تقليل الفترة الزمنية للتنفيذ فعلى سبيل المثال من الممكن عمل مجموعة من rdoQueries في وقت أقل كثيرا عن طريق تضمينهم مع أحد المعاملات.

استخدام أوراكل للتحكم في المعاملات

كما يتضح من هذا الكتاب، فإن مجموعات النتيجة القابلة للتحديث تميل لان يتـــم تقييدهـــا فـــي الاستخدام إلى نظام النهاية المنخفضة ونظام الحجم الصغير ولكن يجد المبرمج نفسه مجبراً علـــى الوصول إلى أعلى موقف من مجموعات نتائج للأمام فقط وللقراءة فقط وفي هـــذه الحالـــة مـــن الممكن تنفيذ عمليات التحديث عن طريق الإجراءات المخزنة.

وتتيح كل تقنيات الدخول إلى البيانات التي تمت مناقشتها في هذا الكتاب تنفيذ عبـــارات SQL من الفيجوال بيسك عبر طريقة Execute وتتم مناقشة هذه الطريقة في كل فصل له علاقة بـــهذا الموضوع وإذا تم استخدام طريقة الإجراء المخزن مع أي من هذه الطرق "أو حتـــى تـــم تنفيـــذ تعليمات DML مثل INSERT, UPDATE, DELETE من الفيجوال بيسك" فإن المستخدم ســـوف يكون أمامه بدائل محدودة للتحكم في المعاملات:

- تضمين عبارة COMMIT في كل إجراء مخزن.
- تمرير إشارة إلى الإجراء المخزن تشير إلى إذا ما كان هذا التغيير سوف يتم من عدمه.
- تنفيذ عبارة COMMIT أو ROLLBACK من الفيجوال بيسك وهو أمر سهل غـــير أنـــه يتطلب رحلة إضافية في الشبكة.
- إذا تم استخدام كتلة تعليمات البرمجية PL/SQL مجهولة الهوية لتحديث البيانـــات، فإنـــه يمكن تضمين عبارة COMMIT مـــع عبـــارة INSERT, أو UPDATE, أو DELETE ويغطي الفصل ١٥ "وحدات البيانات البعيدة" هذا الموضوع.

ويجب على المبرمج أن يفكر جيدا في كيفية التحكم في المعاملات الخاصة به ولا يتم تنفيـــذ قواعد بياناتDAO وأوراكل على النحو الأكمل بعد تنفيذ عبـــارة SQL غـــير أن ADO وRDO على سبيل المثال يتم تنفيذهم عن طريق الاختيار الافتراضي وإذا كانت هناك رغبة في اســـتخدام هذه التقنيات بالإضافة إلى استخدام تقنيات التحكم في المعاملات الموضحة بأعلى فيجب اســـتخدام طريقة Begin Trans قبل تنفيذ بارات DML الخاصة بالمستخدم وبهذه الطريقة يمكن اســـتخدام تحكم واضح في المعاملة كما يمكن أيضا استخدام تقنيات أكثر تعقيدا مـــن أوراكـــل لتضميـــن المعاملات عن طريق استخدام أمر SAVEPOINT SQL.

استخدام خادم المعاملة

التقنية الأخيرة التي يتم استخدامها في التحكـــم فـــي المعاملـــة هـــي اســـتخدام Transaction Processing Monitor. ومن أهـــم خصـــائص برنـــامج فيجـــوال بيســـك هـــو Microsoft Transaction Server (MTS) غير أن هذه الأنظمة كانت لها أهمية عظيمة لســـنوات عديـــدة وكان يعتمد عليها في الحاسبات الإلكترونية الكبيرة مثل CICS.

وإذا تم تطوير أحد التطبيقات ذات الحجم الكبير فيجب أن يتم هذا التطوير عن طريق استخدام أحد المناهج التي تعتمد على الوحدة بالإضافة إلى إنشاء مكونات البرامج التي تحوي عدداً بسيطاً ومترابطاً من الوظائف وفي ما يخص هذه الوجهة فإن وثائق الفيجوال بيسك جيدة جداً. وتقدم مايكروسوفت نوعاً من البنية التي تعتمد على أساس الخدمات وهى لا تختلف عن البنية المتعددة الطبقات ولكن الكلمات الخاصة بهذه الخدمة لن تؤدي إلى ارتباك المستخدمين لأنهم لن يظنوا أن كل طبقة يجب أن تكون على جهاز منفصل.

ويجب أن يكون المستخدم على دراية تامة بأنه هناك عدة مستويات من الخدمة وسوف تتم كتابة المستوى المتوسط من الخدمة في برنامج الفيجوال بيسك كما أنه سوف يتم تشغيله في خادم بيئة النوافذ WINDOWS NT تحت MTS وتتكون طبقات الخدمات من:

١- خدمات المستخدم: واجهة GUI ويجب أن تكون غير معقدة.

٢- خدمات العمل: وهو المكان الذي يتم وضع قواعد العمل فيه.

٣- خدمات البيانات: توفر هذه الخدمات واجهة مصدر البيانات وسوف يتم جمعهما دائما مع خدمات العمل ولكن ليس هناك ضرورة لأن يكونوا هكذا. ومن الممكن لمكونات خدمة البيانات هذه أن تستخدم الإجراءات المخزنة أو أوامر SQL ولكن يتم إدارة التحكم في المعاملة من خلال MTS وليس من التطبيق بشكل مباشر.

ومن الواضح أن أوراكل يسعد بمثل هذه النماذج من الخدمات غير أنه دائما ما تتم كتابة مثل هذه البيانات في PL/SQL كإجراءات مخزنة كما يمكن نقلها إلى خادم آخر تحت تطبيق أوراكل لخادم الويب ويتم كتابة هذه الخدمات في جافا.

ومن الصعب تنفيذ طرق كل من مايكروسوفت وأوراكل المفضلة في الوقت الراهن وبرنامج أوراكل يتوافق مع XA وهو من أحد الضروريات للعمل مع Microsoft Transaction Server غير أنه هناك العديد من الموضوعات التي يجب الفطنة إليها.ورغم أن هذا الكتاب لا يناقش MTS، إلا أنه يوجد هناك بعض الوثائق الخاصة ببرنامج VB.6 التي من الممكن أن تساعد القارئ، كما أنه من الممكن لبرنامج Oracle Application Server أن يساعد برنامج فيجوال بسك "من خلال COM".

موضوعات القفل

يتمتع برنامج أوراكل بنظام قفل متميز وهو الذي تم التعرض له في الفصل الثاني من هذا الكتاب "بنية قاعدة بيانات أوراكل" وهو الذي يعالج المواقف التي قد يواجهها القارئ. وعلى الرغم من ذلك فإنه يجب على القارئ أن يكون على وعي تام عن كيفية تأثير القفل على التطبيق وما هي القرارات الواجب اتخاذها والتي من شأنها أن تؤثر على القفل الموجود بالفعل.

أولاً، نبذة سريعة عن استراتيجية القفل لأوراكل: يتم وضع قفل خاص على الصف أو الصفوف التي تم التعامل معها ويحدث ذلك في أي وقت تجري فيه عملية تحديث أو حذف. ومن الممكن لمستخدمين آخرين قراءة هذه الصفوف غير أنه من غير الممكن تغييرها ألا إذا قام المستخدم الأول بعمل هذه التغييرات وتعتمد المعلومات التي يتم استرجاعها بواسطة مستخدمين آخرين على القيم التي كانت موجودة قبل تحديث وقفل الصفوف وعند اختيار صفوف البيانات من قاعدة البيانات يظهر العرض وهو لا يحتوي على أية صفوف تحتوي على أية أقفال شاملة في أية معاملة غير منفذة في النظام وقراءة أحد الصفوف لا تعني وضع أي نوع من الإقفال على الصفوف.

ويتم جذب كل عمود مختار في كل صف مختار إلى الذاكرة في جهاز المزود قدر الإمكان مع مؤشر للقراءة فقط قابل للإزاحة ويعد هذا Recordset ما هو إلا مؤشر للبيانات ولن يحوي أية تغييرات كان قد تم القيام بها عن طريق مستخدمين آخرين والتي تم تنفيذها بعد إنشاء Recordset الخاص بالمستخدم ويجب إعادة بناء Recordset إذا ما كانت هناك رغبة في رؤية أي تغييرات عادةً ما يتم هذا البناء باستخدام طريقة Refresh. وإذا ما تم اتخاذ منهج القراءة فقط هذا فإن الطريقة الوحيدة لتحديث البيانات هي عن طريق تنفيذ تعليمات SQL أو عن طريق استدعاء الإجراءات المخزنة: وهذه الملامح تسمح بها كل تقنيات الدخول إلى البيانات على الرغم من أن بعضها يسمح بها بطريقة أفضل من البعض الآخر غير أنه لا خلاف على أنه ليس على المبرمج أن يقلق بخصوص القفل فليس هناك وسيلة للتفريق بين الصف الذي قد تم عمل قفل له من غيره.

وهناك بعض الموضوعات التي يجب أن يتم وضعها في الحسبان فيما يخص Recordsets القابلة للتحديث فسوف يتم تخزين كل المفاتيح على صفوف مجموعة النتيجة في ذاكرة المزود. ويتم جذب البيانات الحالية من قاعدة البيانات عن طريق المفتاح المميز مع ODBC عند الدخول إلى صفوف البيانات وهذا لا يؤدي بدورة إلى عرض جيد لقاعدة البيانات "كمؤشر للوقت". بينما يجذب OO4O كل البيانات إلى ذاكرة محلية عند تشغيل الاستعلام مما يؤدي بدورة إلى عرض متميز. وللمبرمج مطلق الاختيار بين الطريقتين.

وسواء إذا ما تم استخدام ODBC أو OO4O فإن أية تغييرات يتم القيام بها في قاعدة البيانات تنعكس أولاً على Recordset عند تنفيذ طرق AddNew أو Edit ثم تصبح دائمة على قاعدة البيانات عند استخدام طريقة Update وعند القيام بأي تغييرات على البيانات يجب الأخذ في الاعتبار ما إذا كانت هناك رغبة في قفل الصف الذي تم التأثير عليه عند تنفيذ AddNew أو Edit وهو ما يعرف بالقفل المرغوب أو إذا ما كانت هناك رغبة في ترك القفل حتى التحديث وهو ما يعرف بالقفل غير المرغوب.

وعادةً ما يكون القفل المرغوب هو نظام القفل الافتراضي كما أن طريقة الدخول إلى البيانات تقوم بهذه المهمة. وهذا النظام هو أيضاً نظام القفل الطبيعي لأوراكل وإذا ما تمت محاولة تحديث أحد الصفوف التي قام أحد الأشخاص الآخرين بقفله فقد يؤدي ذلك إلى تعطيل التطبيق حتى يتم إيقاف القفل ومن الممكن أن يتغير رد الفعل من قبل طريقة الدخول إلى البيانات التي يتم استخدامها ويجب التجربة لان هذا الموقف قد تحول إلى مشكلة.

والقفل غير المرغوب ضد قاعدة بيانات أوراكل أصعب من القفل المرغوب من الناحية التنفيذية وذلك لان القفل غير المرغوب يرتبط بالموقف الذي يقفل فيه Microsoft SQL Server أحد الصفحات "آلتي من الممكن أن تحوي عدة صفوف من البيانات" بمجرد تنفيذ الأمر Edit ولا يوجد في أوراكل قفل للصفحة "وذلك لأنها لا تحتاجه أصلا" كما لا يوجد أي قفل موازي للقفل غير المرغوب وإذا ما تم استخدام Recordset للقراءة فقط فإنه لا يوجد خيار القفل الآلي وإذا ما كانت هناك رغبة في قفل صف معين عند بدأ Edit بدلا من طريقة Update فإنه يتعين أن تتم تلك الخطوة يدوياً، وللقيام بهذه الخطوة يدوياً فانه يجب استخدام تعليمات أكثر تقدماً من SQL وهذه العبارة تدخل إلى واحد أو أكثر من الصفوف وتقوم بتجهيزها لعملية التحديث. وتقوم هذه العبارة بوضع قفل تام على الصفوف لذلك يجب التأكد من عدم استمرار الغلق لفترة طويلة ويتم عرض هذه المعلومة في تعليمات البرمجة ١٣-٤ ويجب تنفيذها تماماً مثل أي أمر SQL عادي. ويجب ملاحظة أنه يتم تضمين خيارNOWAIT لذلك فإن التطبيق لن يتوقف لانتظار قفل شخص آخر حتى يتم إيقافه وبدلا من ذلك فإن الأمر يتم إيقافه فوراً عند وقوع أحد الأقفال.

تعليمات برمجة١٣-٤

```
strSQL = "SELECT * FROM movies WHERE movie_id = 132" & _
" FOR UPDATE NOWAIT
```

يجب الحرص عند التعامل مع الأمر SELECT FOR UPDATE لأنه من الممكن أن يتداخل مع العمل الطبيعي لطريقة عمل الوصول إلى البيانات التي يتم استخدامها ومن الأفضل محاولة تقيد استخدامها بالنسبة RECORDSETS القابلة للقراءة فقط.

ملاحظة يمتلك OO40 قفل غير مرغوب فقط في Oradynasets بالمقارنة مع طريقة قفل ODBC ويتضمن OO40 لأمر Select For Update الداخلي بمجرد تنفيذ طريقة Edit.

استخدام نوع بيانات BLOB

يتم استخدام المصطلح BLOB فيما يخص البيانات على نحو واسع ويفضل أوراكل استخدام المصطلح BLOB عن هذا المصطلح. ويعد التعامل مع الأعمدة الكبيرة في برنامج أوراكل من الأمور الصعبة خاصةً عندما تحتوي تلك الجداول على بيانات ثنائية وقد يبدو التعامل مع الجداول الكبيرة أسهل عند استخدام فيجوال بيسك عن أوراكل ولكن يحتاج كل منهم إلى تخطيط دقيق ولحسن الحظ توجد بعض التقنيات الشائعة بالنسبة لكل طرق قواعد البيانات التي يأتي ذكرها في هذا الكتاب. وهذه هي طرق GetChunk وAppendChunk.

GetChunk: ويقرأ مجموعة من البيانات من جدول العمود.

AppendChunk: وتلحق مجموعة من البيانات إلى جدول العمود.

هناك العديد من الأمثلة الممتازة لاستخدام AppendChunk وGetChunk في ملفات مساعد الفيجوال بيسك والتي يتم تطبيقها على أوراكل من ضمن قواعد بيانات أخرى. وتكمن المشكلة الحقيقية العملية في أنه يتم نقل كميات هائلة من البيانات بين التطبيق وقاعدة البيانات والمتغيرات التي يتم استخدامها لهذا الغرض يجب أن تكون كبيرة على نحو يسمح باحتواء قيمة العمود كله وإذا ما تم التعامل مع BLOBS بنفس الطريقة التي يتم التعامل بها مع الأعمدة العادية فإنه سوف تكون هناك حاجة ماسة لذاكرة كبيرة حتى تحتوي كل هذا الحجم من المتغيرات.

ملاحظة يسمح RDO بتحديد أقصى حجم للعمود الذي من الممكن أن يحوي عنصر rdoQUERY ومن الممكن استعادة هذا الحجم من خاصية BindThreshold لعنصر rdoQUERY.

وفي بعض الأحيان تكون قراءة وكتابة BLOBS أمراً يستحق التتبع.غير أنه يجب معرفة السبب وراء تخزين BLOBS في قاعدة البيانات أولا قبل الدخول إلى هذا النشاط بدون معرفة السبب.هل يتم تخزين الصور في عارض الويب؟ هل هو نظام معظم للملفات؟ حتى أنة هناك BLOBS يتم استخدامها لتخزين التطبيقات للملفات ذات الامتداد EXE.

وفي أحوال عديدة يتم استخدام قواعد البيانات لتخزين هذه الملفات وذلك يحدث لتوافر القدرة على ذلك وبمجرد البدء في استخدام أوراكل فإن المبرمج سوف يجد كل شئ سهلا والبديل لذلك هو تخزين الملفات على أنها ملفات! وإذا كان هناك حاجة للعودة للملفات فإنه يمكن ببساطة تخزين الاسم الكامل المؤهل ويتم ذلك على ملف عام لشبكة خادم ويتم برمجة التطبيق وفقاً لذلك وبهذا يتحقق أداء أحسن بصورة واضحة.

وهذا لا يعني بالطبع عدم تخزين BLOBs في قاعدة البيانات. حيث أن بعض الأدوات التي تقدمها أوراكل تتطلب أن يتم تخزين البيانات في قاعدة البيانات، مثل اختيار Context الذي يتم استخدامه في فهرسة كل كلمة في أحد النصوص، أو اختيارات Image Processing المذهلة التي تسمح بتفحص الصور التي تم تخزينها للألوان والمواد. ويسمح أوراكل بتخزين ملفات كبيرة الحجم جداً من الخارج ويقدم أوراكل ٨ النوع الجديد من البيانات BFILE والذي يسمح بحد أقصى 4GB من البيانات الثنائية ليتم وضعها خارج قاعدة البيانات.

تحذير يجب مراعاة أنه عند التعامل مع البيانات الثنائية أن برنامج فيجوال بيسك يستخدم سلاسل Unicode والتي تأخذ عدد ٢ بايت لكل حرف. وحيث أن برنامج أوراكل يقبل بيانات بتعليمات برمجية من بايت واحد فسوف يؤدي هذا إلى بعض المشاكل.إذا ما تمت محاولة وضع البيانات الثنائية في سلاسل. وبدلا من ذلك يتم استخدام مصفوفات البايت.

معالجة الأخطاء

في أي وقت يتم فيه البدء في تصميم أحد التطبيقات حول قاعدة البيانات، فإن هذا يعني أنه سوف يتم التعامل مع أنواع مختلفة من الأخطاء حتى أن المبرمج يحتاج إلى تحديد منهجاً قياسياً في معالجة الأخطاء والتأكد من أن التطبيق يمكن أن يتوافق مع أي نوع من الأخطاء في أية وحدة نمطية. ويقوم هذا الجزء من الكتاب بتغطية هذا الموضوع.

تصيد الأخطاء

يقدم برنامج الفيجوال بيسك للمبرمج بعض التعليمات القليلة التي تسمح له باختيار شكل التعامل مع الخطأ المراد استدعاؤه. وفي اللغات الأخرى قد :يكون هناك تعليمات برمجية للرجوع والتي تمكن المبرمج من الاختبار والتعامل الخاص الذي يمكن المبرمج من الوصول إلى روتين معين للتعامل مع الأخطاء. ويوفر برنامج الفيجوال بيسك هذين النوعين من التقنيات. وعند الممارسة قد يتم استخدام أكثر من واحدة من هذه التعليمات في إجراء واحد. والأمر متروك للمبرمج لفهم جميع التقنيات وتنفيذها بالشكل الذي يعتقد أنه سوف يعمل على النحو الأمثل. وتعليمات تصيد الأخطاء هي:

ON ERROR GOTO LABEL ويعد هذا هو أكثر تقنيات تصيد الأخطاء المستخدمة انتشاراً في EXEPTION HANDLING والذي يعني الانتقال السريع إلى التعليمات

البرمجية التي تستطيع التعامل مع هذا الخطأ راجع تعليمات برمجة (١٣-٥) وهناك اصطلاح غير مكتوب ينص على أن التعليمات البرمجية الخاصة بمعالجة الأخطاء سوف تكون في نهاية إجراء الوظيفة وعادة ما يسبقها تعليمات EXIT SUB حتى لا تقوم التعليمات البرمجية الخاصة بالمبرمج بتنفيذها.

On Error Resume Next: وهذا يعد بديلاً لتقنية التعامل الاستثنائي EXCEPTION HANDLING التي تغلق تصيد الأخطاء وفي هذه الحالة يقع العبء على المبرمج لفحص ما إذا حدثت أخطاء من عدمه في كل مرة. ويحدث هذا بشكل طبيعي بواسطة عنصر ERR.

On Error Goto 0: وهذه التعليمات غامضة إلى حد ما لأنها تنقل تصيد الخطأ إلى حالته الافتراضية. بعبارة أخرى، لا يتم تصيد أي من الأخطاء بعد هذه التعليمات في الإجراء الحالي وإذا لم يتم التعامل معها فإن ذلك سوف يؤدي إلى توقف التطبيق في النهاية. مع ظهور مربع الرسالة الشهير. وعند الحاجة إلى تمرير أية أخطاء عن عمد إلى إجراءات الاستدعاء الخاصة بهم فإن هذه التعليمات تصبح ضرورية، لأن أي خطأ لا يتم التعامل معه يتم تمريره إلى مجموعة الاستدعاء.

Resume Next: وتواصل هذه العبارة التنفيذ بعد وقوع أحد الأخطاء في السطر التالي للسطر الذي أدى إلى الخطأ. ومن الأفضل أن يتم معرفة المكان الفعلي لوقوع الخطأ عند القيام بتصحيح الأخطاء في التطبيق.

Raise: يجب استخدام طريقة RAISE لإنشاء أحد أحداث الأخطاء ويتم استخدام هذه التعليمات في أحد المكونات عندما لا تتوافر الرغبة في حدوث أي نوع من التفاعل مع المستخدم. وعند رفع أحد الأخطاء الجديدة (في معالجة الأخطاء على سبيل المثال) فإنه سوف يتم تمريرها مره أخرى إلى إجراء الاستدعاء حتى يتم التعامل معها هناك، ويظهر مثال لذلك في تعليمات برمجة ١٣-٥.

Default Behavior: ويستخدم السلوك الافتراضي في أي موقف من مواقف الأخطاء في برنامج الفيجوال بيسك لإبلاغ المستخدم في حالة وقوع أي خطأ عن طريق مربع الرسالة والإغلاق ويجب التأكد من أن هذا لن يحدث لأي من التطبيقات.

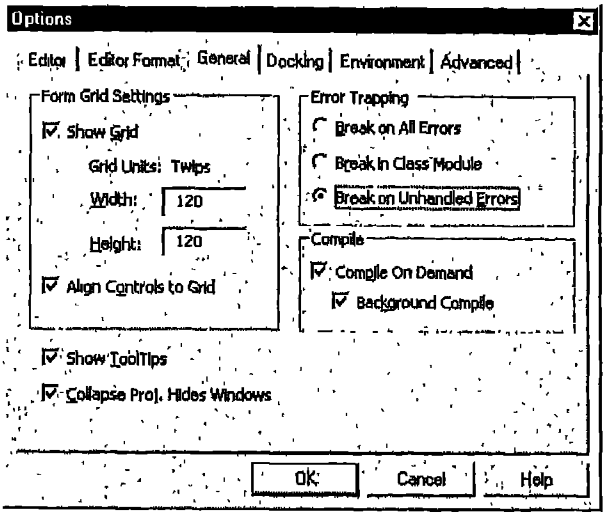
تعليمات برمجة ١٣-٥

```
Private Sub Demo ()
'  Application code
Exit Sub
```

```
ErrorHandler:
Err.Raise Err.Number, "ErrorDemo: Creating Recordset", _
Err.Description
End Sub
```

تصيد الأخطاء في بيئة التطوير

في أحوال عدة قد يستخدم المبرمج عبارة On Error Resume Next بدون أي فائدة ويجب معرفة أنه يمكن ضبط خصائص تصيد الأخطاء في مشروع التغيير. ولأداء هذا الإجراء فإنه يجب الذهاب إلى General من قائمة Tools ⇦ Options وإذا كان هناك Break في All Error المختارة وهذا سوف يؤدي إلى تجاهل كل الأخطاء التي تم تصيدها. راجع شكل (١٣-٥).



الشكل ١٣-٥
يجب التأكد من أن التطبيق لا يقف على كل الأخطاء

تطوير رسالة خطأ Error Message

من أحد التقنيات الشائعة الاستخدام في معالجة الأخطاء هو عرض رسالة بالخطأ الذي حدث متضمنة وصف هذا الخط. وقد تكون هذه الرسالة سهلة مثل الملف المفقود أو قد تكون رسالة معقدة من أوراكل وإذا تم إعطاء المستخدم هذه الرسالة إلى المستخدمين فإن هذا سوف يؤدي إلى ارتباكهم والحل الأفضل هو عرض إحدى الرسائل التي تشير إلى:

١- ما هو الخطأ بصورة مبسطة.

٢- ما هو الإجراء الذي يمكن اتخاذه في هذه الحالة.

٣- إلى من يلجأ إذا تعذر التعامل مع الخطأ.

ومحاولة تجنب حدوث رسائل الأخطاء يعد من أحد المناهج الجيدة في التصميم. وهذا عادةً ما يتطلب تصميما جيداً بوجه عام وأسهل الطرق لذلك هو تقييد المعلومات التي يستطيع المستخدمون إدخالها إلى التطبيق الخاص بالمبرمج ويجب الاستفادة من مربعات القوائم وخانات الاختيار ومربعات الخيار على النحو الأمثل ويتم محاولة تجنب مربعات التحرير والسرد التي تسمح للمستخدم بإدخال السلاسل الخاصة به إذا كان هذا ممكناً. ولكن يجب أيضاً أن يكون من السهل لأي مستخدم إضافة أي قيم جديدة لمربع القائمة قدر الإمكان كما انه لا يجب حذف أي صيغة لإدخال البيانات للتنقل بين سلسلة من قوائم صيانة البيانات.

ويجب مراجعة كيفية دخول المستخدمين إلى الأجزاء المختلفة من التطبيق حتى تتضح كيفية تجنب رسائل الأخطاء. وإذا كانت قواعد العمل تتتبع انتهاك الأمن فإن هذا يعني أنها على استعداد لرفع أحد حالات الأخطاء فعلى سبيل المثال إذا استطاع أحد المستخدمين الدخول إلى صيغ إدخال البيانات الخاصة Human Resources وحاول تحديثها فإن الإجراءات المخزنة سوف تتتبع انتهاك الأمن وترفع حالة الخطأ. ويعد هذا التدقيق الأمني في غاية الأهمية ولكن أليس إيقاف بند القائمة أو زر الصيغة الذي بدأ العملية يعد تحسناً ملحوظاً ؟ولكن لا تزال هناك حاجة إلى اختيار Backend في حالة ما إذا وجد أحد الأشخاص طريقة لمعرفة التعليمات البرمجية ولكن سوف يظهر التطبيق بصورة أكثر جمالاً وسهولة عند الاستخدام. ويجب ألا يكون المستخدم في أحد المواقف التي قد تؤدي إلى انتهاك أحد قواعد العمل.

وتكمن المشكلة الكبرى في تحديد أي العمليات المسموح بها لكل مستخدم. والتقنية المفضلة هي وضع جدول واحد في قاعدة البيانات والتي تعرف كل مستخدم. الامتيازات التي يمكنه الحصول عليها. وعند بداية التطبيق فإنه يتم إجراء فحص لتتبع الامتيازات الخاصة بكل مستخدم ويتم تثبيت الصيغ والقوائم وفقاً لذلك. ومن السرعة إلى حد ما محاولة إيقاف التحكمات والقوائم أو حتى جعلها غير مرئية.

تحذير لا يتم وضع أي معالج للأخطاء يحتاج إلى تداخل المستخدم في أحد المكونات أبداً ويجب التعامل مع الخطأ على نحو كامل في المكون أو رفع الخطأ إلى طبقة واجهة المستخدم.

نوافذ الخطأ أم أشرطة الحالة

هناك محاولات لتجنب استخدام نافذة الخطأ "سواء كانت تعليمات MSGBOX أو صيغة خاصة بالمبرمج" وتبديلها باستخدام تقنيات أخرى لا تتداخل مع انسياب استخدام التطبيق. ومن الغريب أنه من وقت قريب كان خط الحالة هو الطريقة الوحيدة لإخبار المستخدم عند حدوث أي خطأ وإلى حد ما فإن ماسح WEB يقوم بالتحكم في هذه الحالة.

وما تزال الفكرة وراء هذه النقلة إلى أشرطة الحالة منطقية: حيث أن نقر زر OK من نافذة الخطأ قد يكون مزعجاً. كما أن هناك كثير من المستخدمين لا يقرؤون هذه النوافذ.

غير أنه يجب التفكير جيداً قبل إهمال نافذة الخطأ تماماً. حيث أن مستخدمي النوافذ قد يجدوا بعض الصعوبة في ملاحظتهم الدائمة لشريط الحالة ومن الممكن الوصول إلى حل وسط عن طريق وضع الانتهاكات في شريط الحالة وأخطاء النظام في نافذة للخطأ.

تسجيل الأخطاء

رغم أن الأخطاء التي تنتج عن انتهاك قواعد العمل الخاصة بالمستخدم لا تحتاج إلى استرجاع إلا أنه يجب عمل نقطة لإعداد أحد الأنظمة للدخول إلى كل الأخطاء الأخرى التي يعتقد أنها أخطاء خطيرة وهي تلك الأخطاء التي تأتي من أوراكل بالإضافة إلى الأخطاء الداخلية من التطبيق وهي أخطاء البرمجة.

وهناك العديد من الطرق المقبولة لتنفيذ الدخول:

- تسجيل أحد الملفات المحلية على جهاز المستخدم.
- تسجيل أحد الملفات على الشبكة.
- تسجيل ملف على قاعدة البيانات.
- تسجيل Registry الخاص بأحد المستخدمين.
- تسجيل تثبيت نظام مبني على أساس البريد الإلكتروني.

تسجيل البريد الإلكتروني

وجد أن الأنظمة المبنية على نظام البريد الإلكتروني جذابة لكثير من المستخدمين حيث أنها توفر ميزة الإبلاغ السريع عن تفاصيل الخطأ للمبرمج. مما يتيح للمبرمج تصحيح أي خطأ في قاعدة البيانات قبل ملاحظته من قبل المستخدم. كما أن المستخدمين لا يفضلون تقارير الأخطاء لأنهم في هذه الحالة سوف يشعرون بالمسؤولية لتدمير هذا البرنامج غير أن العيب في البريد الإلكتروني يظهر في أنه لا يوفر تسجيلاً للتاريخ لذلك من الأفضل دمجه مع أحد المناهج الأخرى.

وكمية البيانات المراد الدخول إليها عادة ما تكون اختيارية غير أنه من المفيد استخدام القائمة التالية:

١- البيانات والتاريخ عند حدوث الخطأ.

٢- رقم الخطأ.

٣- رسالة الخطأ.

٤- مصدر الخطأ "والذي يعرف التطبيق والوحدة النمطية والإجراء".

٥- اسم مستخدم أوراكل.

٦- رقم إصدار التطبيق.

٧- اسم قاعدة البيانات ورقم الإصدار إذا كان هذا متاحاً.

أخطاء أوراكل

في الجزء المتبقي من الكتاب يتم عرض مجموعات معينة في كل طريقة للدخول إلى البيانات يتم استخدامها للتقرير عن حدوث أي أخطاء من ODBC فعلى سبيل المثال قد تكون مجموعات ERRORS أو rdoERRORS وإذا ما تم استخدام عنصر ERR من الفيجوال بيسك فإن هذا يعني حدوث خطأ جديد ومجموعات الخطأ تحتوى على كل المعلومات المفيدة في أول عنصر خاص ERROR "مثل: Errors (0).Description" وشكل هذا الخطأ المرتد من ODBC هو:

[vendor] [ODBC_component] [data_source] message

وهذا المثال يوضح ذلك:

S0022: [Microsoft][ODBC driver for Oracle][Oracle]ORA-00904: invalid column name

والأمر الهام الذي يجب الفطنة إليه مع ODBC أو OO4O هو أن رسالة الخطأ الخاصة بأوراكل "ORA-00904:Invalid column name" لن تعني الكثير بالنسبة للمستخدم لذلك فإنه يجب إظهار هذه الرسالة بصورة أفضل. وأفضل الطرق للتعامل مع المشكلة هو إخبار المستخدم ببساطة أن هناك خطأ داخلي وأنه يجب عليه طلب الدعم الفني في أقرب فرصة. ويجب التأكد من أن الرسالة ممكن الحصول عليها من التطبيق: من زر "More Information".

تلميح بعض الأخطاء تكون "Application Specific" وتختلف حسب برنامج التشغيل المستخدم ومن الممكن إيجاد أكبر عدد ممكن من رسائل الأخطاء المتوقعة من برنامج تشغيل ODBC قبل إنتاج البرنامج. وينطبق هذا خاصةً على الأخطاء التي تتضمن اعتمادات تسجيل دخول غير صحيحة وتتبع أحد أمثلة أوراكل التي يتعذر الحصول عليها.

خُلاصة

تم تغطية بعض الموضوعات الخاصة بالتصميم الموجودة في الفيجوال بيسك مع أيٍ من طرق الوصول إلى البيانات في هذا الفصل كما أنه تم عرض كيفية إنشاء طريقه أكثر قبولاً للدخول إلى قاعدة البيانات بالمقارنة مع تقديم مربع حوار الافتراضي ODBC Driver Manage فقط، بالإضافة إلى ذلك فقد تمت مناقشة عملية اختيار البيانات من قاعدة البيانات وتحديثها، بما في ذلك التحكم في المعاملة وأي من موضوعات القفل. كما تمت تغطية موضوعات بيانات BLOB-type في قاعدة البيانات والمنهج المقترح لتخزين البيانات على أحد ملفات الخادم. وقد تم إنهاء هذا الفصل بفكرة مختصرة عن معالجة الأخطاء.

ويتم التعرض إلى طرق تسجيل البيانات التي تكون متاحة لمبرمج الفيجوال بيسك في الفصول التالية.

١٤

وحدات الوصول إلى البيانات

وحدات الوصول إلى البيانات هي الاسم المعطى إلى نموذج الوحدات لمحرك JET. وهذا المفهوم متواجد منذ سنوات عديدة ولقد مر بتغيير كثيرة، ولكنه مازال من النماذج المصممة للتعامل مع قواعد البيانات مثل مايكروسوفت أكسس. ولقد تم إضافة إمكانية الوصول إلى مصادر بيانات ODBC "وبدون ذلك لم تكن لتظهر هذه الإمكانية في هذا الكتاب". كما اتبعت DAO منهج فيجوال بيسيك بكونها، مع فيجوال بيسيك ٥، طريقة وصول ٣٢ بت فقط. إذا أردت الوصول إلى قاعدة بيانات ١٦ بت، عليك البحث عن الإصدارات القديمة.

على الرغم من احتواء DAO على أضعف وظيفة موجودة من تقنيات الوصول إلى البيانات، فهي من أشهر التقنيات المعروفة. كما تتواجد أيضا في بيئات أخرى مثل مايكروسوفت أكسل. إذا كان لديك تطبيق جاري تشغيله بصورة مرضية في DAO، فمن المحتمل أنك لن تفكر في تغييره. مع ذلك إذا كنت تقوم بتكوين مشروع جديد، فيجب أن تفكر جديا في استخدام ODBCDirect أو وحدات البيانات عن بعد Remote Data Objects أو من الأفضل وحدات بيانات ActiveX (ADO). إذا كانت هناك حاجة فورية مع توافر مطوري DAO، فمن الممكن أن تكون راضيا عن DAO للوقت الحالي، فيجب أن تتحرك تدريجيا من نموذج الوصول إلى البيانات إلى ADO. سيكون لديك منحنى تعليمي، ولكن الأحوال ستكون جيدة على المدى الطويل.

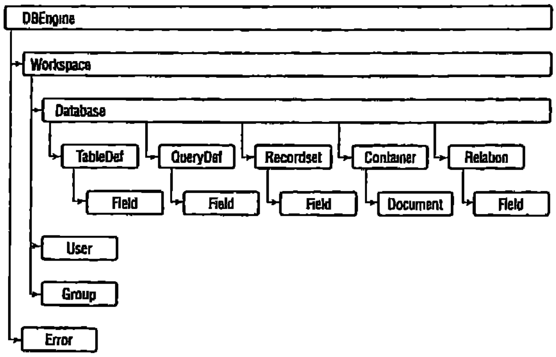
على الجانب الإيجابي، على الرغم من التحديد على نموذج قاعدة بيانات JET، فلا زال هناك وظيفة كبيرة بصورة مدهشة لمستخدم أوراكل في نموذج DAO، ومع الانتقال إلى الإصدار ٣,٥، قد أصبح الأداء في مقبولا بدرجة كبيرة.

تتضمن الموضوعات التي سنناقشها في هذا الفصل ما يلي:

- تدرج وحدة وحدات الوصول إلى البيانات.
- الوصول إلى قاعدة بيانات.
- تكوين مجموعات السجلات.
- تنفيذ أوامر SQL.
- استدعاء الإجراءات المخزونة.
- معالجة الأخطاء.
- التجميع.
- عرض واستخدام هيكل قاعدة بيانات.
- عيوب DAO.

تدرج وحدة وحدات الوصول إلى البيانات

سوف نشاهد عبر الفصول المتبقية من هذا الكتاب، بينما نشاهد الطرق المتنوعة للتداخل مع أوراكل من خلال فيجوال بيسيك، منهج على أساس الوحدة بالنسبة للأساليب التي تمت مناقشتها. وهذا يعني وجود تدرج من الوحدات الأخرى وتجميعاتها، كل بالخصائص والطرق الخاصة به، أسفل وحدة بمستوى أعلى من نموذج DAO، وهو الأمر الذي يسمح لك بالوصول إلى البيانات الموجودة في قاعدة البيانات. لقد أثبت النموذج القائم على أساس الوحدة سهولة في الاستخدام والقابلية للتأقلم. في حالة DAO، يتم عرض تدرج الوحدة في الشكل ١٤-١. يتم حينئذ بيان الوحدات ذات المستوى الأعلى في التدرج بصورة فردية.



الشكل ١٤-١

كائنات الوصول إلى البيانات. تدرج الكائنات

DBEngine

الوحدة ذات المستوى الأعلى في تدرج وحدة وحدات الوصول إلى البيانات هي DBEngine. من الممكن وجود وحدة DBEngine واحدة فقط، وستحتوي على كل الوحدات الأخرى في التدرج. لا يتم تكوين وحدة DBEngine، حيث أنها متاحة بصورة افتراضية ولكن لا يمكنك تكوين واحدة أخرى. الوحدات الموجودة أسفل DBEngine هي تجميعات من الوحدات، ولذا فإن الأخطاء، على سبيل المثال، هي مجموعة من وحدات الأخطاء. تحتوي وحدة DBEngine على ملفات الجداول وتجميعات الأخطاء.

ملف الجداول

يتركز معظم تدرج DAO أسفل تجميع ملفات الجداول. ملف الجداول هو عبارة عن هيكل تنظيمي يسمح لك بتجميع اتصالات قاعدة البيانات داخل منطقة تم تسميتها أو جلسة عمل. ستحتاج إلى اكثر من ملف جدول إذا أردت الوصول إلى مجموعات قواعد البيانات بطرق مختلفة، مثل مجموعة معينة من خلال محرك teJ أو من خلال ODBCDirect آخر (والذي سيتم

مناقشته في الفصل السادس عشر، ODBCDirect). ولكن على الرغم من ذلك، فمن المحتمل أن تكون سعيدا باستخدام ملف الجداول الافتراضي الذي يسمى ملفات الجداول (٠) كما سوف نشاهد في القسم التالي. يتم فتح ملف الجداول هذا تلقائيا بمجرد الإشارة إليه ويظل مفتوحا حتى نهاية التطبيق. تحت محرك teJ، تحتوي وحدة ملف الجداول على مجموعات ومستخدمين ومجموعات قواعد بيانات.

تلميح إذا كنت تقوم بالوصول إلى قاعدة بيانات واحدة، فيجب أن تستخدم ملف الجداول الافتراضي فقط. تحافظ هذه الخطوة على قصر وسهولة صيانة الرمز.

المجموعات والمستخدمون

في قاعدة بيانات Jet، مثل مايكروسوفت أكسس، من الممكن تحديد مجموعات العمل التي تتكون من مجموعات ومستخدمين والأذونات المرتبطة بهم بغرض الوصول إلى قاعدة البيانات. تستخدم هذه الوحدات بصورة أساسية في قواعد بيانات أكسس والذي يتم تخزين الأذونات في ملف system.mdw. لا تحتوي المجموعات والمستخدمين أية قيمة بالنسبة لمستخدم قاعدة بيانات أوراكل.

قاعدة البيانات

تعتبر وحدة قاعدة البيانات من أهم الوحدات التي سنتعامل معها حيث أنها تسمح لك بالتوسيط مع قاعدة البيانات وعرض هيكلها واسترداد وتحديث مجموعات السجلات منها وتنفيذ أوامر SQL مقابل قاعدة البيانات. تشكل وحدة قاعدة البيانات الأساس بالنسبة لبقية هذا الفصل.

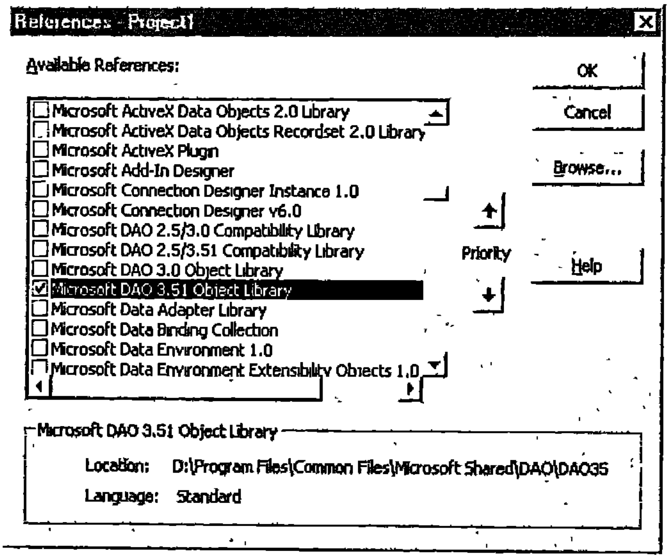
الوصول إلى قاعدة بيانات

من الممكن أن تكون قد لاحظت أنه إذا تم اختيار مشروع Exe قياسي في فيجوال بيسك، يتم تواجد مراقبة البيانات في المشروع بصورة افتراضية، مجرد نقرة واحدة ويمكنك وضعه داخل نموذج. إذا كنت تستخدم فيجوال بيسيك ٦، لا قتم باختيار مشروع البيانات حيث أن هذا سيؤدي إلى إضافة مرجع إلى مكتبة ADO. لاستخدام مكتبة وحدة الوصول إلى البيانات، ستحتاج إلى مزيد من العمل. يجب إضافة مرجع المكتبة إلى المشروع. إذا لم تقم بتنفيذ ذلك، لن يتم تشغيل الرمز. لإضافة المرجع، استخدم الخطوات الآتية:

١- اختر مشروع، مراجع لفتح مربع حوار المراجع الموجود في الشـــكل ١٤-٢. ســوف تشاهد قائمة بكل مكتبات الوحدات التي يمكن الرجوع إليها في التطبيق.

٢- حدد الخيار المعنون مكتبة وحدة مايكروسوفت DAO ٣,٥١ (أو ٣,٥ أو فيجوال بيسـيك ٥).

٣- انقر موافق.



الشكل ١٤-٢
مربـــــع حـــوار Refrence التي تم اختيارها.

عند إضافة مراجع، تعتبر مكتبة DAO متاحة للمشروع. تعرض هذه المكتبـــة كـــل الطـــرق والخصائص التي تحتاجها لتكوين ملفات الجداول وفتح قواعـــد البيانـــات وتكويـــن مجموعـــات سجلات.

أول شي سنقوم به هو فتح قاعدة بيانات تخزين الفيديو المثال. لتنفيذ هذه الخطوة، يتم استخدام طريقة OpenDatabase. تعتبر هذه الطريقة جزء من وحدة ملف الجداول، ولكن من الممكـــن استخدم ملف الجداول الافتراضي بدون الانشغال بتكوين ملف جديد. تقوم هذه الطريقـــة بإعـــادة مرجع إلى وحدة قاعدة بيانات، ولذا يجب استخدام أمر إعداد. يقوم هذا الأمر بتخصيص مرجـــع وحدة إلى متغير وحدة نوع قاعدة البيانات.

صياغة أمر OpenDatabase هي كالتالي:

Set *database* = _
workspace.OpenDatabase (dbname, options, read-only, connect)

وتحتوي طريقة OpenDatabase على المعاملات الآتية:

dbname "اسم قاعدة بيانات": اسم مصدر البيانات DSN لقاعدة بيانات أوراكل الذي تم إعداده في مدير مشغل ODBC. يتم تقديم هذا المعلومات عادة في خاصية التوصيل، ولذا يمكنك ترك هذا الحقل خاليا.

Options خيارات: قم بإعداد هذا المعامل على خطأ الافتراضي إذا أردت فتح قاعدة البيانات في وضع خاص، والبديل هو الوضع المشترك. في الواقع يشير هذا الخيار إلى قاعدة بيانات JET، وبالنسبة لقاعدة بيانات أوراكل، يجب تحديد خطأ بصورة دائمة. لاحظ أنه بالنسبة لقاعدة بيانات ODBCDirect، والذي سوف نناقشه في الفصل السادس عشر، توجد مجموعة مختلفة من الخيارات.

read-only "قراءة فقط": قم بإعداد هذا المعامل على صواب إذا أردت أن تكون قاعدة البيانات للقراءة فقط، وإلا قم بإعداده على خطأ، وهو الافتراضي، بحيث يمكنك القراءة والكتابة على قاعدة البيانات. يعتبر هذا النوع من السرية من الأمور الخاصة بالتطبيق ويجب تقديمه بدون إحلاله محل سرية أوراكل.

Connect "توصيل": هذه هي سلسة التعليمات التي تحتاجها للتوصيل إلى قاعدة بيانات أوراكل. هذه السلسة سيتم مناقشتها في الفصل الثاني عشر، إمكانية الوصول إلى أوراكل باستخدام فيجوال بيسيك. والآن من الممكن تجميع هذه المعلومات مع الاستمرار في التوصيل إلى قاعدة البيانات كما هو معروض في تعليمات برمجة ١٤-١.

تعليمات برمجة ١٤-١

```
Private dbMovies As Database
Private Sub ConnectToDatabase ()
'  Declare and open a database using the default workspace.
 Set dbMovies = Workspaces(0).OpenDatabase _
     ("", False, False, "ODBC;DSN=movies;")
End Sub
```

أين تريد الاحتفاظ بوحدة قاعدة البيانات

عند تكوين وحدة قاعدة بيانات، ستريد إتاحة تلك الوحدة في كل البرنامج. من التمرينات القياسية في تطبيقات المزود الخادم هو فتح اتصال بقاعدة البيانات عند بدأ

تشغيل التطبيق وإغلاق الاتصال عند إغلاق التطبيق. في تعليمات برمجية١٤-١ تجاهلنا هذه المسألة بمجرد تحديد وحدة قاعدة البيانات، dbMovies، بنفس الإجراءات التي تم تكوين الوحدة بها. في الحقيقة هناك ثلاث طرق مختلفة على الأقل لمعالجة مكان وضع وحدة قاعدة البيانات:

- حدد وحدة قاعدة البيانات كمتغير عام في الوحدة النمطية للبرنامج، ثم قم بالرجوع إليها في كل البرنامج. هذا هو من أبسط المناهج.
- اتبع منح توجيه الوحدة وقم بتخزين وحدة قاعدة البيانات في متغير خاص في طبقة، والتي من الممكن الرجوع إليها بواسطة طبقات أخرى.
- قم بفتح وإغلاق اتصال قاعدة البيانات عند الحاجة، من الممكن أن تكون هذه الخطوة من الخطوات المكلفة ولكنها من الممكن أن تفيد إذا أردت عمل اتصالات قليلة وقم بإعداد الخادم بحيث تكون الاتصالات سريع "من المحتمل باستخدام عمليات الخادم قبل التشغيل أو مجمع الاتصال". يعتبر هذا من المناهج النموذجية بالنسبة لمكونات الطبقة المتوسطة التي تتطلبها الخدمة من خادم ويب. تعتبر مصروفات DAO الإضافية عالية بدرجة كبيرة بالنسبة لهذه الطريقة. إذا كان هذا هو الحل الذي تريده فيجب أن تستخدم وحدات بيانات ActiveX، والذي من الممكن أن يكون وقت الاتصال أسرع.

خصائص وحدة قاعدة البيانات

بجرد الاتصال بقاعدة بيانات أوراكل، يمكنك الوصول إلى طرق عديدة للحصول على مزيد من المعلومات. من المحتمل عدم استخدامها بصورة كبيرة، ولكن من الممكن أن تريد الحصول على مزيد من التفاصيل حول كيفية استخدام اتصال قاعدة البيانات. تم عرض أكثر الخصائص الشائعة في الجدول ١٤-١.

الجدول ١٤-١: خصائص وحدة قاعدة البيانات.

الخاصية	البيان
Connect "توصيل"	سلسلة عمليات التوصيل الفعلي المستخدم.
Name "الاسم"	اسم قاعدة البيانات كما هو مقدم في طريقة OpenDatabase.
QueryTimeout "انتهاء	فترة انتهاء الوقت بالثواني بالنسبة لأي استعلام.

الجدول ١٤-١: خصائص وحدة قاعدة البيانات.

الخاصيـة	البيان
وقت الاستعلام"	
RecordsAffected "السجلات المتأثرة"	عدد السجلات المتأثرة بواسطة أحدث طريقة تنفيذ في قاعدة البيانات. لا تقدم هذه الخاصية معلومات مفيدة لمستخدمي أوراكل.
Transactions "الحركات"	تبين ما إذا كانت الحركات مسموح بها في قاعدة البيانات أم لا.
Updatable "قـــــابل للتحديث"	يبين ما إذا كانت الحركات قابلة للتحديث أم لا.

تكوين مجموعات السجلات

والآن بعد أن قمنا بفتح وحدة قاعدة بيانات لتطبيق تخزين_فيديو، نحتاج فيما بعد إلى التمكن مـن الوصول إلى البيانات من قاعدة البيانات. على سبيل المثال، من الممكن أن نريد تشغيل اسـتعلام لاسترداد كل أشرطة الفيديو المعروضة للإيجار. عند قيام أوراكل بتنفيذ تطبيق مثل هـذا، فإنـه يتيح مجموعة النتائج لتطبيق فيجوال بيسيك. في DAO، تسمى الوحدة التي نحتاج إلـى تكوينـها لقبول هذه البيانات باسم Recordset (مجموعة سجلات). يتم تكوين وحدة مجموعـة سـجلات جديدة باستخدام طريقة OpenRecordset (مجموعة سجلات مفتوحة) من وحدة قاعدة البيانـات. تضيف الطريقة مجموعة سجلات جديدة إلـى تحصيـل مجموعـات السـجلات. يقـوم أمـر OpenRecordset بتكوين مرجعا إلى وحدة مجموعة سجلات جديدة، ولذا يجب تخصيصه إلـى متغير مجموعة سجلات باستخدام أمر Set.

صياغة طريقة OpenRecordset هي كالتالي:

```
Set recordset = _
  database_object.OpenRecordset (source, type, options, lockedits)
```

تحتوي الطريقة على المعاملات التالية:

source "المصدر": هذا هو سلسلة العمليات الذي يمثل مصدر بيانات مجموعة سجلات جديـدة. سوف يكون جملة SQL نموذجية محاطة بعلامة اقتباس مزدوجة، مثـل "SELECT * FROM

videos WHERE status='A'". لا تضع فاصلة منقوطة في نهاية السطر. لاحظ أنه لا يمكنك استخدام اسم جدول أوراكل في هذا الحقل.

Type "النوع": ثابت اختياري يشير إلى نوع مجموعة السجلات المراد فتحها، كما هو محدد في الجدول ١٤-٢.

Options "الخيارات": توليفة اختيارية من الثوابت تحدد خصائص مجموعة السجلات الجديدة. يمكنك استخدام واحدا أو أكثر من الثوابت المحددة في الجدول ١٤-٣. إذا كنت تستخدم أكثر من ثابت يمكنك إضافتها معا. تم تحديد ثوابت إضافية عديدة في ملفات التعليمات، ولكن هذه الطريقة لا تعمل مع أوراكل.

Lockedits: ثابت اختياري يحدد عملية تأمين مجموعة السجلات. يمكنك اختيار ثابت من الثوابت المحددة في الجدول ١٤-٤.

الجدول ١٤-٢: الثوابت الممكنة لمعامل النوع من طريقة OpenRecordset.

الثابت	البيان
DbOpenDynaset	يقوم هذا الثابت بفتح وحدة مجموعة سجلات من نوع dynaset والتي تعتبر قابلة للتحديث. وهي شبيهة بمؤشر مجموعة مفاتيح ODBC
DbOpenSnapshot	يقوم هذا الثابت بفتح وحدة مجموعة سجلات من نوع لقطة، والذي يعتبر غير قابل للتحديث. وهي شبيهة بمؤشر ODBC ثابت.
DbOpenForwardOnly	يقوم هذا الثابت بفتح مؤشر للأمام فقط، والذي، كما يشير اسمه، يقوم بدعم طريقة MoveNext فقط. ينتج عن طرق النقل الأخرى خطأ ممكن الوقوع فيه.
DbOpenTable	يقوم هذا الثابت بفتح وحدة مجموعة سجلات من نوع الجداول (غير متاح أوراكل).

الجدول ١٤-٣: الثوابت الممكنة لمعامل الخيارات من طريقة OpenRecordset.

الثابت	البيان
DbAppendOnly	يسمح هذا الثابت للمستخدمين القيام بإلحاق سجلات جديدة إلى Recordset ولكن بدون تحرير أو إضافة سجلات موجودة (مجموعات سجلات قابلة للتحديث فقط).
Dbsqlpassthrough	يقوم هذا الثابت جملة SQL SELECT مباشرة إلى أوراكل، متجاهلا معالج استعلام JET. يقوم هذا بتكوين Recordset للقراءة فقط، ولكن ستحتاج إلى استخدام خيار dbSQLPassThrough إذا أردت استخدام صياغة خاصة بأوراكل، متضمنة للوظائف والمخططات.
DbInconsistent	يشير هذا الثابت إلى إمكانية تحديث Recordsets من روابط الجدول مثل عدم استمرار فعالية شرط ربط. لا تعتبر Recordsets من روابط الجدول في أوراكل قابلة للتحديث، ولذا يعتبر هذا الثابت غير قابل للاستخدام.
DbConsistent	يشترط هذا الثابت وجوب بقاء شروط ربط الجداول في Recordsets قابلة للتحديث. ثانيا يعتبر هذا الثابت غير قابل للاستخدام مع أوراكل.

الجدول ١٤-٤: الثوابت الممكنة لمعامل Lockedits من طريقة OpenRecordset.

الثابت	البيان
DbReadOnly	يمنع هذا الثابت المستخدمين من عمل أية تغييرات في مجموعة السجلات Recordset. على الرغم من قيام أوراكل بمراقبة ما إذا كنت ستقوم بتبديل البيانات، بصورة أساسية، يسمح لك هذا الخيار بفرض للقراءة فقط. كما يعني أيضا وجود مصروفات إضافية أقل للوصول إلى البيانات. يمكنك استخدام dbReadOnly إما في

الجدول ١٤-٤ : الثوابت الممكنة لمعامل Lockedits من طريقة OpenRecordset.

الثابت	البيان
	معامل الخيارات أو معامل lockedits ولكن ليس في كليهما. إذا حاولت استخدامه في كلا المعاملين، سيظهر خطأ مشغل.
DbOptimistic	يحتوي هذا الثابت على تأمين مرغوب، والذي يتضمن عدم إقفال الصفحة (السطر في حالة أوراكل) والتي تحتوي على السجل حتى يتم تنفيذ طريقة التحديث. يعتبر هذا هو الافتراضي بالنسبة لمصادر ODBC، مثل أوراكل.
DbPessimistic	يحتوي هذا الثابت على تأمين غير مرغوب ، والذي يتضمن أن الصفحة الموجود بها السجل مقفلة عند تنشيط طريقة Edit (التحرير). لا تعمل طريقة التأمين غير المرغوبة في مقابل قاعدة بيانات أوراكل.

ملاحظة تم تبسيط جداول قيم المعاملات المذكورة مسبقا لإزالة أية خيارات غير قابلة للتطبيق بالنسبة لأوراكل. في توثيق فيجوال بيسيك، ستجد مجموعة متنوعة من الثوابت الأخرى والتي تنطبق على قواعد بيانات JET.

عند قيامك بتكوين Recordset من قاعدة بيانات أوراكل، سوف يكون القرار الرئيسي الواجب اتخاذه هو ما إذا كنت تريد جعل Recordset قابلة للتحديث أو للقراءة فقط. بمجرد اتخاذ قرار بخصوص ذلك، يوجد هناك نوعين من طريقة OpenRecordset تحتاج إلى أخذهم في الاعتبار، وسوف نشاهد كليهما في الأقسام القليلة الماضية.

تكوين مجموعات سجلات قابلة للتحديث

مجموعات السجلات القابلة للتحديث هي عبارة عن مجموعات سجلات يمكنك قراءتها والكتابة عليها. بالطبع لجعل ذلك محل تنفيذ، ستحتاج إلى تحديد وإضافة وتحديث وحذف الميزات في كل الجداول التي تريد الوصول إليها. ستعتاد على العمل مع هذه الأنواع من مجموعات السجلات إذا كنت متمرسا للعمل مع مايكروسوفت أكسس أو استخدمت مراقبة البيانات. تبين رموز التعليمات التالية كيفية تكوين مجموعة سجلات قابلة للتحديث.

```
Dim rs as Recordset
Set rs = dbMovies.OpenRecordset _
        ("SELECT * FROM [video_store.movies]", dbOpenDynaset)
```

يقوم هذا المثال بفتح مجموعة سجلات جديدة قائمة على أساس استعلام SQL.

```
SELECT * FROM [video_store.movies]
```

لاحظ عدم إضافة فاصلة منقوطة في نهاية جملة SQL. تذكر أن أية تغييرات في صفوف البيانات الممثلة في مجموعة السجلات بواسطة مستخدمين آخرين لن تظهر في مجموعة السجلات حتى يتم تنشيطها. تتمثل المشكلة الوحيدة مع هذه العملية فى إمكانية قيام مستخدمين آخرين بتغيير نفس السجلات التي تريد تغييرها أو أسوء من ذلك وهو إمكانية حذف السجلات من قاعدة البيانات.

لاحقا في هذا الفصل في قسم "التجميع"، سوف نناقش مثال نموذج مع مجموعة سجلات قابلة للتحديث.

تلميح أحد المشاكل الرئيسية مع استخدام مجموعات سجلات قابلة للتحديث مع DAO هي قيام JET بمعالجة طلبات الاستعلام الخاصة بك، دون التعرف على صياغة أوراكل. في حالة مجموعات السجلات القابلة للتحديث، إذا استخدمت مخطط تطبيق (وهذا بالتأكيد ما ستفعله)، إذا ستحتاج إلى إرفاق أسماء الجداول في أقواس معقوفة مثل [video_store.movies].

تكوين مجموعات سجلات للقراءة فقط

البديل لمجموعات السجلات القابلة للتحديث هو مجموعات سجلات للقراءة فقط، ولكن لماذا تختار شيئا لا يمكنك تحديثه؟ هناك ثلاث ميزات رئيسية لمجموعة السجلات للقراءة فقط:

- أنها أكثر كفاءة من وجهة نظر الخادم والمزود.
- يمكنك استخدام خيار SQLPassThrough بالإضافة إلى استخدام صياغة أوراكل، بما في ذلك أسماء المخططات.
- أنها تؤدي إلى بنية اكثر قابلية للتحجيم.

تنبع كفاءة كون مجموعة السجلات للقراءة فقط من حقيقة السهولة الشديدة لمؤشر ODBC المستخدم، بدون الحاجة إلى إمكانية التحديث. مع ذلك، بينما يتم تخزين كل البيانات الموجودة في مجموعة سجلات على جهاز المزود في الذاكرة، يجب التأكد من إضافة الأعمدة التي تحتاجها

بصورة مطلقة في جملة SELECT فقط. إذا كان بإمكانك استخدام مؤشر التوجيه فقط، فسوف تكتسب كفاءة أكثر على حساب القدرة على التنقل ضمن البيانات مع ذلك يمكنك تكرار استخدام المؤشر بنفسك بصورة دائمة وفعالة، على سبيل المثال في شبكة للقراءة فقط.

تلميح على الرغم من الميزات النظرية للمؤشرات البطيئة في DAO، فلقد اكتشفت، في اختبارات تمت على الآلاف من الصفوف، أنه من الصعب قياس نسبة الأداء الفعلي. الهدف من هذه المسالة هو التجريب باستخدام البيئة الخاصة بك.

مع اللقطات، يجب أيضا ممارسة الحذر لعدم تضمين بيانات من أعمدة LONG RAW "صفوف طويلة" أو من أعمدة من نوع BLOB. تذكر أن المؤشر يقوم بقراءة كل البيانات الموجودة في الذاكرة، ومن الممكن أن تكون هناك نتائج مدمرة إذا تم عمل ذلك مع كميات كبيرة من البيانات الثنائية. إذا سمح لك التطبيق بذلك، يجب أن تقوم بالوصول إلى الحقول الطويلة فقط عند الحاجة. كمثال عل هذا التطبيق هو عند قيامك بعرض بيانات عن موظف ولكن مع القيام فقط بالوصول إلى صورة عن موظف معين عند نقر الزر المصاحب. يقصد بتضمين عمود الصورة كجزء من مجموعة السجلات أنه يتم سحب كل الصور داخل الذاكرة. لا تعتبر هذه مشكلة إذا كانت هناك سجلات قليلة فقط في مجموعة السجلات، ولكنها من الممكن أن تصبح مشكلة موارد خطيرة. في هذه الأحوال، من الممكن أن يكون فعلا من الأمور الأكثر فعالية قيامك باستخدام مؤشر قابل للتحديث.

ملاحظة للحصول على مناقشة خاصة بمؤشرات ODBC، راجع الفصل الحادي عشر، تقنية جانب المزود.

على الرغم من ميزات مجموعات سجلات للقراءة فقط، لا يمكنك تحديث السجلات الموجودة بها، ويعتبر ذلك جيدا في ظروف معينة "على سبيل المثال، حيثما تريد ملء مربع قائمة بيانات منضم مستخدم لخيارات إدخال البيانات"، ولكن بخلاف ذلك يتطلب استخدام مجموعات سجلات للقراءة فقط منهج مختلف تماما لتصميم التطبيق. إذا اتخذت منهج الواجب تنفيذ كل الإضافات والتحديثات وعمليات الحذف في قاعدة البيانات بواسطة تنفيذ جمل DML أو، على نحو أفضل، فإن منهج القراءة فقط هو الطريق الذي يجب أن تسلكه. في كثير من البيئات، وبخاصة تجاه النهاية العليا، ببساطة لا يوجد لديك خيار مؤشر قابل للتحديث، ويجب استخدام طريقة مفردة وأكثر فعالية متاحة.

ملاحظة في الفصل الخامس عشر "وحدات البيانات البعيدة" عن بعد، سوف نشاهد أساليب تسمح لك في الواقع بتحديث البيانات باستخدام جمل DML أو إجراءات مخزونة مع انعكاس التغييرات في نفس الوقت في مجموعة النتائج القابلة للتحديث.

لاحقا في الفصل الموجود في قسم "التجميع"، سوف نشاهد مثال للنموذج مع مجموعة سجلات للقراءة فقط يسمح بعمليات التحديث من خلال الإجراءات المخزونة.

منهج شخصي في التعامل مع Recordsets

في حالة التطبيقات الخاصة بي شخصيا، فإنني أميل إلى استخدام منهج القراءة فقط لمجموعات السجلات. من الممكن عرض البيانات المستردة في نموذج فيجوال بيسك بطرق عديدة مثل مربعات النص والحوار والسرد أو من المحتمل في شبكة أو تحكم جدول بيانات. عند قيام مستخدم بتحديث أية بيانات، فإن هذا التحديث يتم بواسطة استدعاء الإجراءات المخزونة. سوف نستعرض كيفية القيام بهذا لاحقا في هذا الفصل.

لا يعتبر هذا من الصعوبات المعينة، إذا كنت تستخدم وحدات تحكم بيانات منضمة، يجب ترميز الإجراءات لتنفيذ ذلك على أية حال. يمكنك فقط مع مجموعات السجلات القابلة للتحديث استخدام طرق AddNew "إضافة جديد" وedit "تحرير" وUpdate "تحديث" من مجموعة السجلات.

بمجرد كتابة الإجراءات المخزونة، يصبح لديك نسبة تحكم أكبر على العملية. بالنسبة لهؤلاء الأشخاص الذين يتخوفون من مهمة كتابة الإجراءات المخزونة لإجراء الإضافة والتحديث والحذف في كل جدول في قاعدة البيانات، سوف نقوم بتطوير برنامج DAO للمساعدة في أتمتة هذه العملية في نهاية الفصل.

من الممكن استخدام هذه المعلومات عمليا وتكوين مجموعة سجلات قراءة فقط. فيما يلي جزء عينة من الرمز:

```
Dim rs as Recordset
Set rs = dbMovies.OpenRecordset ("SELECT * FROM
video_store.movies", _
        dbSQLPassThrough, dbOpenSnapshot)
```

ستلاحظ وجود ثابت dbOpenSnapshot في معامل lockedits كما هو متوقع، ولكن أيضا هناك ثابت dbSQLPassThrough في معامل الخيارات. يتضمن ثابت dbSQLPassThrough تجاوز أمر SQL المستخدم لمصدر مجموعة السجلات للمعالج استعلام JET واتجاهه مباشرة إلى أوراكل. لا يقوم هذا فقط بتجاوز أي تحليل يريد JET القيام به في الاستعلام، بما في ذلك روابط الأداء المحتمل ثانية، ولكن من الممكن استخدام صياغة أوراكل بدلا من صياغة JET. يعتبر ذلك مفيدا عندما نريد استخدام وظائف أوراكل، مثل TO_DATE لتحويل التواريخ إلى صياغة أوراكل. كما تعني أيضا أنه من الممكن استخدام المخططات في الاستعلامات. ولكن العيب الوحيد في هذا الخيار هو أن مجموعة السجلات سوف تصبح الآن للقراءة فقط ولكن لا ضرر في ذلك على الإطلاق حيث أننا نريد ذلك على أية حال.

سواء كنا نستخدم مجموعات سجلات قابلة للتحديث أو للقراءة فقط، فإن مشغل ODBC لأوراكل المقدم مع فيجوال بيسيك ٦ يسمح لنا باستخدام مرادفات وطرق عرض عامة وخاصة. تعتبر هذه الخاصية من التحسينات الفعلية بالنسبة لبعض المشغلات القديمة والتي تواجه بعض الصعوبات مع هذه الإمكانية. مع ذلك فمن الواضح فقدان القدرة على معالجة حزم PL/SQL. إذا كان هذا من الأمور المهمة بالنسبة لعملية التطوير الخاصة بك، فمن الممكن أن يكون من الواجب التبديل إلى وحدات بيانات ActiveX من وحدات أوراكل بالنسبة لـ OLE (ربط الكائنات وتضمينها).

صياغة محرك JET

لدى معالج استعلام JET الصياغة القريبة الخاصة به بالنسبة لكثير من أوامر SQL (مثل الروابط ومحددات التاريخ). في كثير من الحالات يمكنك استخدام صياغة JET بدلا من صياغة أوراكل والاستمرار بدون خيار dbSQLPassThrough. مع ذلك يجب تعلم لغتين من SQL. كما يمكنك استخدام مسلسلات الحروف الموجودة في محلل ODBC لجعل الاستعلام الخاص بك استعلام خاص بقاعدة بيانات، ولكن هذه الطريقة تبدو مشكلة بالنسبة للشخص المقتنع بأداء أوراكل للأشياء.

خصائص مجموعة السجلات

بمجرد قيامك بفتح مجموعة سجل، هناك خصائص عديدة يمكنك الوصول إليها. بعض من هذه الأمور الأكثر أهمية تم عرضها في الجدول ١٤-٥. سوف نشاهد كيفية استخدام الكثير منها في القسم التالي.

الجدول ١٤-٥: خصائص مجموعة سجلات محددة.

الخصائص	البيان
AbsolutePosition	تقوم هذه الخاصية بنفل مجموعة السجلات إلى رقم سجل معين.
BOF	تعتبر هذه الخاصية صحيحة إذا قمت بالنقل قبل السجل الأول في مجموعة السجلات.
Bookmark	تشير هذه الخاصبة إلى سجل في مجموعة السجلات.
Bookmarkable	تشير هذه الخاصية إلى إمكانية استخدام الإشارات المرجعية في مجموعة السجلات هذه.
CacheSize	تشير هذه الخاصية إلى حجم الذاكرة الوسيطة للسجل. الافتراضي هو صفر.
CacheStart	تشير هذه الخاصية إلى [illegible] المرجعية الخاصة ببداية الذاكرة الوسيطة للسجل.
EditMode	تشير هذه الخاصية إلى حالة تحرير السجل الحالي.
EOF	تعتبر هذه الخاصية صحيحة إذا [illegible] بالنقل بعد السجل الأخير في مجموعة [illegible]
LockEdits	تعكس [illegible] الخاصية معامل LockEdits في طريقة OpenRecordset.
Name	هذه الخاصية هي سلسلة عمليات المصدر التي [illegible] تتكون منها مجموعة [illegible]
NoMatch	تشير هذه الخاصية [illegible] إلى عدم وجود مطابقة [illegible] البحث.
RecordCount	[illegible] في مجموعة السجلات. هذه الخاصية [illegible] ذات معنى حتى يتم ملء مجموعة السجلات بصورة تامة، على سبيل المثال، باستخدام طريقة MoveLast.

الجدول ١٤-٥: خصائص مجموعة سجلات محددة.

الخصـائص	البيان
Transactions	تشير هذه الخاصية إلى إمكانية دعم مجموعة السجلات للحركات.
Type	تشير هذه الخاصية إلى قيمة معامل النوع الموجودة في طريقة OpenRecordset.
Updatable	تشير هذه الخاصية إلى إمكانبة قابلية مجموعة السجلات للتحديث.

استكشاف طرق مجموعة السجلات

بمجرد قيامك بفتح مجموعة سجلات، هناك طرق عديدة يمكنك الوصول إليها. يمكنك استخدام هذه الطرق للتحكم ففي كيفية سحب البيانات من الخادم ولتغيير القيم في مجموعة السجلات وللتنقل خلال مجموعة السجلات وللبحث عن المعايير الخاصة بمطابقة السجلات.

ملء مجموعة سجلات

أحد الأشياء التي من غير الممكن أن تفكر بها مطلقا في الماضي هي عملية القيام بملء مجموعة السجلات. الملء هو العملية التي تحدث عند سحب الصفوف من خادم أوراكل إلى داخل مجموعات السجلات. يتم تنفيذ ذلك بمعد سطر واحد في المرة الواحدة، دون الكل معا. يتم استرداد الصفوف عند الوصول إليها ولكن ليس قبل طلبها. كنتيجة لذلك، من الممكن أن تتطلب العملية جولات كثير على الشبكة مع كل أوجه القصور المرافقة.

على الرغم من الاستمرار في استخدام الطريقة الافتراضية لملء مجموعة سجلات، لديك أيضا خيار المزيد من التحكم. توفر DAO الآن ذاكرة وسيطة تقوم بتخزين السجلات في مخزن مؤقت في الذاكرة على جهاز المزود، مع ذلك تعمل هذه الطريقة مع dynasets فقط. يمكنك ملء هذه بواسطة التحديد الصريح لعدد السجلات المراد استردادها في المرة الواحدة باستخدام خاصية CacheSize ثم القيام بالملء الفعلي للذاكرة الوسيطة باستخدام طريقة FillCache. الخاصية الأخرى الجديرة بالملاحظة هنا هي خاصية CacheStart والتي تحدد إشارة مرجعية في مجموعة السجلات. تقوم طريقة FillCache بملء الذاكرة الوسيطة بداية من السجل ذا الإشارة المرجعية.

كمثال على ملء الذاكرة الوسيطة، يقوم رمز التعليمات التالي بإعداد الإشارة المرجعية للسجل الحالي ثم إعداد حجم الذاكرة الوسيطة على ١٠٠ سجل ثم في النهاية ملء الذاكرة الوسيطة حتى ١٠٠ سجل.

```
With rsMovies
    .CacheStart =.Bookmark
    .CacheSize = 100
    .FillCache
End With
```

على الرغم من أن هذه الطريقة ستعمل فقط مع مجموعات السجلات القابلة للتحديث، هناك احتمال آخر عند الرضا عن وضع للقراءة فقط. يتمثل هذا البديل في استخدام طريقة GetRows، والتي سيتم مناقشتها فيما بعد في هذا الفصل. مع كلتا هاتين الطريقتين، من الممكن توقع حدوث زيادة معقولة في سرعة ملء مجموعة السجلات. يعتمد التحسين على متغيرات كثيرة، ولكن من الممكن زيادة سرعة البيانات إلى الضعف.

تغيير السجل الحالي

عند تكوين مجموعة سجلات في البداية، يتم وضعها بصورة دائمة في السجل الأول. هناك أربعة طرق مستخدمة بصورة نموذجية للتنقل ضمن مجموعة سجلات قابلة للتمرير كما هو معروض في الجدول ١٤-٦. لاحظ أنه بالنسبة لمجموعة السجلات للتوجيه فقط، يمكنك استخدام طريقة MoveNext فقط. يمكنك تنفيذ هذه الطرق من رموز التعليمات ولكن معظم التطبيقات تقدم وسيط GUI (الرسومي) في شكل أزرار.

الجدول ١٤-٦: طرق نقل مجموعة السجلات.

الطريقة	البيان
MoveFirst	تقوم هذه الطريقة بنقل السجل الحالي إلى السجل الأول في مجموعة السجلات.
MoveLast	تقوم هذه الطريقة بنقل السجل الحالي إلى السجل الأخير في مجموعة السجلات.
MoveNext	تقوم هذه الطريقة بنقل السجل الحالي إلى السجل التالي في مجموعة السجلات.
MovePrevious	تقوم هذه الطريقة بنقل السجل الحالي إلى السجل السابق في مجموعة السجلات.

عند استخدام هذه الطرق، يجب أن تكون على دراية بنتائج الوصول إلى نهاية مجموعة السجلات وكون مجموعة السجلات فارغة. مع MoveNext، يجب أن تقوم بفحص خاصية EOF (نهاية ملف) ومع MovePrevious، يجب فحص خاصية BOF (بداية ملف). إذا قمت بتجربة طرق النقل هذه عند كون EOF أو BOF صحيحة، من الممكن أن تنتهي بسجل حالي غير صحيح.

ملاحظة إذا كان كل من EOF وBOF صحيحا، فإن مجموعة السجلات تكون خالية.

إذا قمت بالنظر على خاصية RecordCount، سوف تشاهد قيمة، ولكن من المحتمل أن هذه القيمة لن تعكس بدقة رقم السجلات الموجود في مجموعة السجلات. لن تكون القيمة عددا موثوقا به حتى تنتقل إلى نهاية مجموعة السجلات باستخدام طريقة من الطرق مثل MoveLast.

الوصول إلى حقول مجوعة سجلات

سواء كنت تستخدم مجموعة سجلات قابلة للتحديث أو للقراءة فقط، ستحتاج إلى تفحص البيانات والتقاط حقول معينة. لتنفيذ ذلك يجب تفهم وحدة مجموعة السجلات بصورة أكثر قليلا. كما هو واضح من الشكل ١٤-١، تحتوي وحدة مجموعة السجلات نفسها على وحدات أخرى عديدة، أحدها هي مجموعة الحقول. الحقل هو عبارة عن تخطيط JET لعمود أوراكل. للبحث عن قيمة لأي حقل في السجل الحالي من مجموعة السجلات، يمكنك الإشارة إليه بطريقة من طريقتين: بواسطة الفهرس (عدد ترتيبي) أو حقول (٠)أو حقول ("اسم").

تلميح كبديل ذلك ولكنه نموذج صياغة أقل إيحاءا للوصول إلى حقل بواسطة الاسم هو الحقول! [الاسم].

عند تجميع هذه المجموعة معا لملء مربع قائمة، يمكننا استخدام رموز التعليمات المعروضة في تعليمات برمجية ١٤-٢. نعم، هناك طرق بيانات منضمة لتنفيذ هذه الخطوة، أيضا، ولكن تعليمات برمجية هذه توضح هذه النقطة.

تعليمات برمجة ١٤-٢

```
Public sub FillListBox (ListBox as control)
'  Fill a list box with movie names
'  Ignore error trapping
   Dim rs As Recordset
   Set rs = dbMovies.OpenRecordset _
     ("SELECT movie_name FROM video_store.movies",
dbOpenSnapshot, _
     dbSQLPassThrough)
   ListBox.Clear
   While NOT rs.EOF
      ListBox.AddItem rs("movie_name")
      rs.MoveNext
   Wend
   ListBox.ListIndex = 0  ' Display the first item
   Set rs = Nothing
End Sub
```

إذا كانت هناك أية إمكانية لكون الحقل NULL خاليا، يجب الاهتمام بشرط الخطأ المحتمل بواسطة التأكد من معالجة المشكلة. بالنسبة لحقل الحرف، يجب إحداث تعاقب الحقل إلى سلسلة فارغة وبالنسبة لقيمة عددية، يجب إضافة سلسلة فارغة ثم تحويلها إلى عددية.

```
strName = "" & rsMovies("Movie_Name")
```

أو

```
iMovieID = Val("" & rsMovies("Movie_Id"))
```

الاختلافات بين JET وأوراكل

شيء واحد يجب وضعه في الاعتبار وهو الاختلاف بين الطريقة التي يردي أن يعيد JET بها الحقول إليك والطريقة التي سيعيدها بها أوراكل عند استخدامك لخيار dbSQLPassThrough. يصبح هذا الاختلاف مهما عندما تقوم بإعادة الحقول من أكثر من جدول باستخدام شرط الضم. إذا قمت باستخدام مؤشر قابل للتحديث (أي

التنقل ضمن JET لاسترداد سجلات بيانات)، حينئذ يتم إعادة كل عمود من الأعمدة المستخدمة لربط الجداول معا في شرط الربط مع اسم العمود الخاص بها مسبوقا باسم الجدول ومفصولا بنقطة. على سبيل المثال، إذا كانت جملة SOURCE (مصدر) مجموعة السجلات هي:

```
SELECT * FROM movies, videos
WHERE movies.movie_id = videos.movie_id
```

سيقوم JET بإعادة حقل يسمى movies.movie_id وكذا واحدا يسمى videos.movie_id. لن تحتوي الحقول الأخرى على سابقة اسم جدول، ولذا سوف تكون movie_name، معدل، وهكذا.

والآن عندما يكون لديك مجموعة سجلات للقراءة فقط مع استخدام خيار dbSQLPassThrough، سوف يقوم أوراكل بإعادة الحقول التي يريد إعادتها. في هذه الحالة، سوف تحصل على ملف ضم واحد movie_id. إذا كنت بحاجة إلى كليهما لأي سبب كان، حينئذ يجب استخدام بدائل الأعمدة كما هو معروض في الفصل ٧ "الاستعلامات".

ولذا كن حذرا حيث أن هاتين الطريقتين من الممكن أن تعطيا أسماء حقول مختلفة وأيضا عددا مختلفا من نفس الاستعلام.

تغيير السجلات والقيم في مجموعة سجلات

هناك طرق قليلة يمكنك استخدامها لتغيير السجلات وقيمها في مجموعة السجلات، على افتراض أنك تستخدم مؤشر قابل للتحديث. أولا يجب التنقل إلى السجل المراد تغييره باستخدام طرق النقل ثم استخدام طرق التحرير التالية:

AddNew: إضافة سجل جديد إلى مجموعة السجلات دون أوراكل. للجعل التغيير دائما يجب استخدام أمر UPDATE (التحديث).

Edit: وضع السجل الحالي في طور التحرير أي من الممكن تغيير أي من قيمه.

Update: حفظ أية تغييرات تمت على خادم أوراكل، على افتراض مطابقة كل الثوابت.

Delete: حذف السجل الحالي. لا تستخدم طريقة التحديث بعد الحذف.

إذا قمت بتنفيذ طريقة AddNew أو Edit، يمكنك تغيير أي من الحقول الموجودة في مجموعة السجلات. يتم تنفيذ ذلك كما يلي:

rsMovies ("Name") = "The Wonderful World of Oracle"

سوف يقوم JET أولا بتنفيذ المراجعة الخاصة به. لا يعتبر هذا الإجراء من الإجراءات المعقدة، ولكنه سوف يقوم، على سبيل المثال، بالتحقق من عدم قيامك بتخصيص سلسلة حروف إلى حقل رقمي. إذا جربت شيئا كذلك، سوف تقع في خطأ شركي "خطأ تحويل نوع البيانات".

عندما تقوم بكل التغييرات التي تريدها من المحتمل بواسطة الإشارة إلى حقل نص في نموذج الإدخال، يمكنك استخدام طريقة التحديث. إذا كانت هناك أية مشاكل في القيود، مثل مفتاح أو معدل غير صحيح ليس جزءا من القائمة المحددة، سوف تحصل حينئذ على خطأ شركي.

إذا كان صف أوراكل الذي تحاول تحديثه مقفل بالفعل بواسطة مستخدم آخر، فسوف يتم توقف التطبيق عن العمل حتى يتم إزالة القفل. لا يقوم محرك JET بمعالجة هذا الموقف بصورة لائقة، وإذا كان هذا الأمر سيسبب مشكلة فمن الممكن التفكير في استخدام صياغة SELECT FOR UPDATE NOWAIT التي شاهدناها في الفصل الثالث عشر، موضوعات تصميم فيجوال بيسيك. وهذا هو المكان الذي تريد وضع الإقفال الحقيقي.

عند إضافة سجل جديد إلى مجموعة السجلات، يجب التأكد من احتواء كلل الحقول المطلوبة. أي، تلك التي تحتوي على قيد NOT NULL على قيمة صحيحة بالفعل. من الممكن أن تكون عملية البحث عن هذه القيم مشكلة، خاصة إذا كنت تستخدم مسلسلات للمفتاح الأساسي. يجب في الواقع قراءة قيم المفتاح من قاعدة البيانات باستخدام جملة SELECT قبل تحديث مجموعة السجلات، ولكن هذه الخطوة من الأمور المزعجة. مع ذلك، يمكنك تنفيذ هذه الخطوة بواسطة تمرير زناد على الجدول، ولكن تنفيذ ذلك من الممكن أن يكون أكثر فوضى مع ترك مجموعة السجلات الخاصة بقيمة غير صحيحة. ولكن إذا اقتصرت على مجموعات سجلات للقراءة فقط، استخدم الإجراءات المخزونة أو قم بتنفيذ رمز SQL لتحديث البيانات ثم تجديد مجموعة السجلات، سوف يكون لديك نظام عام أفضل.

البحث عن سجل معين

آخر شيء سوف نبحث عنه لطرق مجموعة السجلات هو كيفية البحث عن سجل يطابق المعايير الموضوعة. هناك أربعة طرق وخاصية واحدة تحتاج إلى معرفتها، وهي معروضة في الجدول ١٤-٧.

الجدول ١٤-٧: الطرق والخصائص المستخدمة للبحث عن السجلات.

الطريقة	بيان الخاصية
FindFirst	تقوم هذه الطريقة بالبحث عن أول ظهور لعنصر البحث في مجموعة السجلات.
FindLast	تقوم هذه الطريقة بالبحث عن آخر ظهور لعنصر البحث في مجموعة السجلات.
FindNext	تقوم هذه الطريقة بالبحث عن الظهور التالي لعنصر البحث بعد السجل الحالي.
FindPrevious	تقوم هذه الطريقة بالبحث عن الظهور السابق لعنصر البحث قبل السجل الحالي.
NoMatch	تشير هذه الخاصية إلى عدم وجود أية مطابقة لأوامر البحث.

يمكنك استخدام هذه الطرق بسهولة للواسطة إعطاء عنصر البحث. لاحظ أنه يجب أن يكون عنصر البحث في صياغة JET. بعيدا عن ذلك القيد، يمكنك استخدام نفس المعايير بصورة دائمة كما هو الحال في عبارة WHERE من استعلام SELECT. يوضح المثال التالي هذه النقطة.

```
rsMovies.FindFirst "rating = 'PG-13'"
If rsMovies.NoMatch Then
   MsgBox "No matching records were found for the rating PG-13"
End If
```

يجب أن تلاحظ أن أوامر البحث هذه ليست بديلا عن تكوين استعلام SELECT مكتوب بشكل جيد في المقام الأول. إذا استخدمت FindNext وFindPrevious،فإن هذا الاستخدام يعتبر كإشارة إلى أنه من المحتمل القيام بدمج معايير البحث داخل استعلام مصدر مجموعة السجلات.

إغلاق مجموعة سجلات

يتم إغلاق مجموعة السجلات آليا بمجرد خروجها عن النطاق، مثال ذلك، إذا قمت بتحديدها في حدث Form_Load، سيتم إغلاقها بمجرد الخروج من حدث Form_Load. مع ذلك إذا قمت بإعلان مجموعة السجلات كمتغير مستوى نموذج، فمن الممكن أن تجد أنها لن تخرج عن النطاق

ومن الممكن تركها مفتوحة. من التمارين الجيدة القيام إغلاق مجموعة السجلات بمجرد الانتهاء منها. يضمن هذا تحرير مساحة SQL الخاصة في خادم أوراكل وأية ذاكرة مستخدمة بواسطة ODBC بأسرع وقت ممكن. لإغلاق مجموعة السجلات هذه تحتاج إلى إعدادها على "Nothing" ما يلي:

```
Set rs = Nothing
```

كبديل لذلك، استخدم طريقة الإغلاق:

```
rs.Close
```

كما شاهدنا في قسم "ملء مجموعة السجلات" لدى أوراكل طريقة استرداد الصفوف من نوع توجيه فقط من مساحة SQL الخاصة في الخادم. يتم تحرير هذه الذاكرة فقط عند قراءة السطر الأخير أو عند إغلاق مجموعة السجلات. إذا احتجت إلى الاحتفاظ بمجموعة السجلات مفتوحة، فيجب الانتقال إلى السجل الأخير في طريقة GetRows.

استخدام طريقة GetRows

عند استعراض عملية ملء مجموعة السجلات، شاهدنا أن DAO، تقوم، بصورة افتراضية، بقراءة سجل واحد في المرة الواحدة كما هو مطلوب. كما شاهدنا كيفية ملء مجموعة السجلات بصورة واضحة باستخدام طريقة FillCache لتحقيق قراءة فعالة لأكثر من صف في المرة الواحدة. يجب أن نقوم الآن باستعراض طريق أخرى حيث أنها فعالة بصورة خاصة ولكنها لا تقوم بالفعل بقراءة البيانات داخل مجموعة سجلات. تقوم طريقة GetRows من وحدة مجموعة السجلات بملء ما يسمى بمصفوفة المتغير بواسطة قراءة عدد كبير من الصفوف من أوراكل جميعا في المرة الواحدة.

مصفوفة المتغير ليست مصفوفة عناصر بيانات المتغير ولكنها نوع بيانات متغير مفرد. تم تصميم المتغيرات لاتخاذ أية أنواع من البيانات سواء كانت أعداد صحيحة أو سلاسل أحرف أو أنواع أكثر تعقيدا. عند استخدامها، تقوم أنظمة التحويل الضمنية بتغيير نوع البيانات لمطابقة الحالة الموجود بها على افتراض أنها ممكنة. على سبيل المثال، إذا قمت بتخصيص متغير إلى متغير عدد صحيح، يتم تحويل قيمة المتغير إلى نوع عدد صحيح. يسمح لنا هذا التحويل بتحديد مصفوفة داخل نوع بيانات مفرد. من الممكن أن تبدوا هذه كنقطة جيدة للمناقشة، ولكن من المهم ملاحظة: عدم قيام متغير المصفوفة بتغليف النتائج في رزمة أنيقة ولكن من الممكن أيضا تمرير مصفوفات النتائج بين المكونات، حتى لو قمت باستخدامها في شيء يشبه خادم حركة مايكروسوفت. "يجب فقط القيام بترتيب متغير واحد على DCOM".

ملاحظـــة حيث أن مصفوفة المتغير لا تستخدم مؤشرا، لذا تعتبر غير قابلة للتحديث.

لاستخدام طريقة GetRows، يجب أولا تقرير كم عدد الصفوف المراد استردادها في المــرة الواحدة. يمكنك استرداد جزء من مجموعة النتائج بالكامل أو المجموعة بالكامل. يمكنــك طلــب صفوف اكثر مما هو موجود، وسف يتم جلب تلك الصفوف الموجودة فقط. إذا قمت بطلب جــزء من مجموعة النتائج، يمكنك جلب المزيد بصورة متكررة حتى لا يتبقى شيء. سوف يقــوم DAO آليا بتحجيم مصفوفة المتغير مع السماح لك باسترداد عدد من الصفوف والأعمدة الممثلـــة فــي المصفوفة. بمجرد ملء مصفوفة المتغير، فمن السهل التكرار داخلها واسترداد القيـــم المطلوبـــة. تعطي التعليمات البرمجية 14.3 مثالا عن تكوين مجموعة السجلات وســـحب البيانـــات داخـــل مصفوفة المتغير ثم التكرار داخل القيم. ولغرض التمثيل، سوف نقوم بســـحب البيانـــات داخـــل فيجوال بيسيك في هيئة كتل من ١٠٠ سجل. مع وجود جهاز بإمكانيات كافية، يمكنـــك، لحســـن الحظ، الوصول إلى الآلاف من السجلات في المرة الواحدة لتحقيق زيادة حقيقية في الأداء.

تعليمات برمجية ١٤-٣

```
Private Function ReadData() As Integer
'  Read in the data into a variant array
   Dim varRecords As Variant  ' The variant array
   Dim iNumRecords As Integer  ' The number of records
   Dim iNumCols As Integer     ' The number of rows
   Dim iRecNo As Integer       ' Record counter
   Dim iColNo As Integer       ' Column counter
   Dim rsMovies As Recordset
   Dim strSQL As String

   strSQL = "SELECT * FROM video_store.movies ORDER BY movie_id"
   Set rsMovies = dbMovies.OpenRecordset(strSQL, dbOpenDynaset)
   varRecords = rsMovies.GetRows(100)
   iNumRecords = UBound(varRecords, 2) + 1
   iNumCols = UBound(varRecords, 1) + 1
'  Process each record
```

```
For iRecNo = 0 To iNumRecords – 1
   Process each column
   For iColNo = 0 To iNumCols – 1
      MsgBox varRecords(iColNo, iRecNo)
   Next
Next
End Function
```

تلميح للحصول على أفضل كفاءة ممكنة من طريقة GetRows، يجب تكوين مجموعة سجلات تحتوي على تلك السطور والأعمدة التي تريدها بالفعل. إذا أردت عدد قليل فقط من الصفوف، فيمكنك البحث عن طريقة أخرى أسرع.

تنفيذ أوامر SQL

من الممكن أن يتماشى استخدام مجموعات السجلات القابلة للتحديث إلى اللحد الذي تريده في الوقت الحالي، ولكن هناك حالات من الممكن أن تريد فيها تنفيذ بعض تعليمات رموز SQL تقوم بتنفيذ أكثر نشاطاً. بلغة مايكروسوفت هناك "Action Queries" استعلامات الإجراءات وهي عبارة عن جمل SQL تقوم بإضافة أو تحديث أو حذف البيانات. بلغة أوراكل، هذه هي جمل Data Manipulation Language (DML)، لغة معالجة البيانات، التي شاهدتها في الفصل الثامن، تحديث قاعدة البيانات.

لتنفيذ جملة SQL من فيجوال بيسيك، سنستخدم طريقة التنفيذ من وحدة قاعدة البيانات كما هو معروض في الصياغة التالية:

```
database_object.Execute source, options
```

معاملات طريقة التنفيذ هي:

source: سلسلة SQL التي تقوم بتحديد جملة DML مفردة المراد تنفيذها.

Options: أحد الثوابت أو كليهما التي تحدد ميزات طريقة التنفيذ كما هو معروض في الجدول ١٤,٨. كشيء متوقع، يجب استخدام خيار dbSQLPassThrough واحد على الأقل.

الجدول ١٤-٨: ثوابت طريقة التنفيذ.

الثابت	البيان
DbSQLPassThrough	يقوم هذا الثابت بتنفيذ استعلام SQL تمريري على سبيل المثال، واحداً يتم تمريره إلى معالج استعلام أوراكل، متجاوزاً محرك JET.
DbFailOnError	يقوم هذا الثابت بسحب التحديثات عند ظهور أي خطأ. من الأفضل إدارة فشل حركة باستخدام الإجراءات المخزونة إذا كان من الممكن.

مثال على أوامر تنفيذ SQL معروض في تعليمات برمجية.١٤-٤.

تعليمات برمجية ١٤-٤

```
Private Sub DeleteMovie(iMovieId As Integer)
'   Delete a movie
    On Error GoTo ErrorHandler
    dbMovies.Execute "DELETE FROM video_store.movies WHERE
id=132", _
                    dbSQLPassThrough
    Exit Sub
ErrorHandler:
    MsgBox "Error deleting movie id = " & iMovieId & " : " & _
            Errors(0).Description
End Sub
```

تلميح لا تقم بوضع فاصلة منقوطة في نهاية جملة SQL، حتى مع خيار SQLPassThrough.

في أي وقت يتم فيه تنفيذ جملة DML، سوف تصبح جزءاً من الحركة. ليس هناك تثبيت آلي للتغييرات عند استخدام هذه الطريقة، لذا كن مستعداً لتثبيتها بوضوح. مع لك فذلك من السهل، كما هو معروض في التعليمات البرمجية ١٤,٥. لسوء الحظ، لا يمكنك توليف أوامر SQL في نفس المكالمة.

تعليمات برمجة ١٤-٥

```
Public Sub Commit()
'  Commit any outstanding updates in the database

   On Error GoTo ErrorHandler
   dbMovies.Execute "Commit", dbSQLPassThrough
   Exit Sub
ErrorHandler:
msgBox "Error committing changes : " & Errors(0).Description
End Sub
```

تحذير لا تقم بمزج التنفيذ الواضح للجملة COMMIT باستخدام أوامر حركة DAO: BeginTrans و CommitTrans حيث أن هذين النوعين من أوامر مراقبة الحركة سوف تكون على اتصالات مختلفة.

استدعاء الإجراءات المخزونة

سوف تريد في النهاية تنفيذ إجراءات أوراكل المخزونة من داخل فيجوال بيسيك. إذا اعتقدت أنه بإمكانك استدعاء إجراء مخزون باستخدام صياغة تنفيذ SQL التي شاهدناها في القسم الأخير، سوف تكون على صواب بصورة جزئية. مع ذلك هناك بعض التحديدات وحيلة صغيرة يجب أن تكون على دراية.

تقوم طريقة التنفيذ من وحدة قاعدة البيانات بتنفيذ كتلة تعليمات برمجية مجهولة، أي كتلة من التعليمات البرمجية غير مخزونة باسم. لتنفيذ إجراء مخزونة، يجب التفاف الاستدعاء في جمل BEGIN و END كما شاهدناه في استعراض PL/SQL في الفصل العاشر، تكوين الإجراءات المخزونة. يقوم هذا بتحويل استدعاء الإجراء المخزون إلى كتلة تعليمات برمجية PL/SQL

المجهولة. مثال عملي على هذا معروض في التعليمات البرمجية ١٤,٦. تقوم هـــذه التعليمـــات البرمجية بتنفيذ الإجراء المخزون movies_insert2 في مخطط video_store. يجب أولاً بناء سلسلة SQL تحتوي على الأمر المراد تنفيذه. يجب أن يشتمل هذا على كل المعاملات.

تعليمات برمجة ١٤-٦

```
Private Sub MovieInsert(iMovieId As Integer, strMovieName As String,
_
        iLength As Integer, strRating As String, iYear As Integer)
'  Insert a new movie into the database

   On Error GoTo ErrorHandler
   Dim strSQL As String
   strSQL = "BEGIN video_store.movies_insert2 ("
   strSQL = strSQL & iMovieId & ", '" & strMovieName & "', "
   strSQL = strSQL & iLength & ", '" & strRating & "', " & iYear
   strSQL = strSQL & "); END;"
   dbMovies.Execute strSQL, dbSQLPassThrough
   Exit Sub
ErrorHandler:
   MsgBox "Error inserting movie : " & Errors(0).Description
End Sub
ErrorHandler:
   MsgBox "Error inserting movie : " & Errors(0).Description
End Sub
```

نوع سلسلة SQL الذي يعطيه هذا هو:

```
BEGIN video_store.movies_insert (321,
  'Retirement Home for Software', 55, 'R', 1995); END;
```

تم عرض الإجراء المخزون والذي تم تنفيذه بالفعل في التعليمات البرمجية ١٤-٧.

تعليمات برمجة ٣-٢

```
PROCEDURE movies_insert2(
  p_movie_id  IN NUMBER,
  p_name      IN VARCHAR2,
  p_length    IN NUMBER,
  p_rating    IN VARCHAR2,
  p_year      IN NUMBER) IS
/*  Insert a new movie into the movie table
    If the formal parameter, p_movie_id is zero it is assigned
    from the next available sequence number
*/
v_movie_id  NUMBER(6);
BEGIN
  IF p_movie_id = 0 THEN
    SELECT movie_sequence_id.NEXTVAL INTO v_movie_id FROM dual;
  ELSE
    v_movie_id := p_movie_id;
  END IF;
  INSERT INTO movies VALUES (
    v_movie_id, p_name, p_length, p_rating, p_year, USER, SYSDATE);
  COMMIT;
EXCEPTION
/* Whatever the problem just rollback and
   pass the error back to Visual Basic */
  WHEN OTHERS THEN
    ROLLBACK;
    RAISE;
END movies_insert2;
```

تلميح إذا كنت تستخدم أرقام مسلسلة للمفاتيح الأساسية، حينئذ يجب تكوين إجراء مخزون يقوم بإدارة عملية استرداد قيمة مسلسل الإجراء التالي. في هذه الحالة، لا تحتاج إلى تمرير قيمة مفتاح إلى الإجراء، مع ذلك فأنا شخصياً أفضل تمرير. للإشارة إلى وجوب قيام الإجراء بتكوين رقم مسلسل. على الرغم من عدم ضرورة هذا في DAO، فهو يمهد الطريق إلى استرداد المفتاح المخصص باستخدام تقنيات أخرى.

يجب أن تقوم بصورة دائماً، تقريباً، بتضمين اسم المخطط في سلسلة SQL في هذه الحالة، استخدمنا video_store. سوف يكون اسم هذا المخطط بصورة طبيعية هو مالك جداول القواعد ومن المحتمل خاص بالتطبيق. للتمكن من تنفيذ الإجراء المخزون، ستحتاج بصورة واضحة إلى الحصول على امتيازات للتنفيذ ذلك. إذا لم تكون لديك ميزة التنفيذ، سوف تشاهد رسالة خطأ مضللة:

```
[Microsoft][ODBC driver for Oracle][Oracle]ORA-06550: line 1, column 7:
PLS-00201: identifier 'VIDEO_STORE.MOVIES_INSERT2' must be declared
ORA-06550: line 1, column 7:
PL/SQL: Statement ignored
```

كما من الممكن أيضاً ظهور رسالة الخطأ إذا قمت بتحديد اسم خطأ للإجراء المخزون.

لا توجد هناك طريقة لإعادة معاملات مخرجات، كما لا يمكنك تنفيذ الإجراءات المخزونة. إذا كنت تريد الحصول على وظيفة اكثر تقدماً، يجب استخدام أحد الطرق المحددة في الفصول الأخيرة من هذا الكتاب.

ملاحظة تقوم وحدات الوصول إلى البيانات بتكوين اتصال ثان أو جلسة عمل إلى أوراكل لكي يقوم بتنفيذ جمل SQL بما في الإجراءات المخزونة. من الممكن أن يكون هذا الأمر من الأمور المربكة للمستخدمين وجود التطبيق في وضع الإقفال بينما يتم تكوين الاتصال الثاني، ولذا إذا كان من الممكن، يجب فرض تكوين هذا الاتصال عند بدأ التشغيل. باستطاعة أي استدعاء لأمر COMMIT القيام بهذه المهمة.

معالجة الأخطاء

بالإضافة إلى وحدة الخطأ المألوف، تزودنا DAO بمجموعة الخطاء الخاصة بالأخطاء المتعلقة بالبيانات. ستحتاج إلى هذه المجموعة لإعطاء معنى للأخطاء المعادة من أوراكل، وإلا فسوف تجد رقم الخطأ المألوف ٣١٤٦. تتكون مجموعة الأخطاء من وحدات الأخطاء التي تم تحديد خصائصها في الجدول ١٤,٩. تحتوي كل وحدة خطأ على معلومات، ولكن الوحدة المفيدة فقط هي الوحدة الأولى الموجودة في المجموعة، أخطاء (٠).

الجدول ١٤-٩: خصائص وحدة الخطأ بالنسبة لأخطاء DAO.

الخاصية	البيان
Number	الرقم الصحيح (عدد صحيح طويل).
Description	سلسلة تحتوي على بيان الخطأ.
Source	اسم الوحدة التي أنشأت الخطأ.
HelpFile	اسم الملف بالكامل لملف تعليمات فيجوال بيسك.
HelpContext	معرف سياق ملف التعليمات لرقم الخطأ.

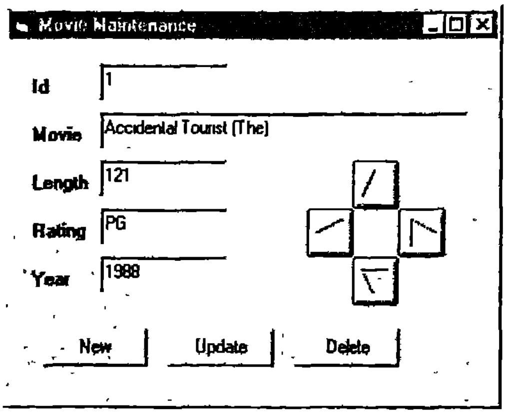
يعطي الفصل ١٣، إصدارات تصميم فيجوال بيسيك، نظرة عامة على ما هو متوقع من أخطاء أوراكل المستردة في ODBC وكيفية استخدامها. سوف تستخدم مربع رسالة وخصائص البيان بصورة نموذجية إما في مربع رسالة أو من المفضل في نظام معالجة الأخطاء.

التجميع

لقد شاهدنا، في الأقسام السابقة، مجموعة متنوعة من التقنيات والخصائص والطرق، سوف نستخدم في هذا القسم هذه المعلومات للتقديم نموذج صغير يسمح لنا بعرض وتحديث قاعدة بيانات أفلام. سوف نأخذ منهجين مختلفين على نحو أساسي لتنفيذ هذا الإجراء. في المثال الأول، سوف نستخدم مجموعة سجلات قابلة للتحديث تعتبر طريقة نموذجية لاستخدم نموذج مايكروسوفت أكسس. في المثال الثاني، سوف نستخدم مجموعة سجلات للقراءة فقط واستدعاء الإجراءات المخزونة للتنفيذ التحديثات. يعتبر المنهج مناسبا للتطبيقات الصغيرة في حين يصبح المنهج الثاني ضرورة في حالة زيادة حجم التطبيق. يستخدم كلا الحلان نفس النموذج كما هو معروض في الشكل ١٤-٣.

الشكل ١٤-٣

نموذج تحديث قاعدة البيانات Movie



التعليمات البرمجية المشتركة في البرنامج معروضة في التعليمات البرمجية ١٤-٨. سوف نستخدم هذه التعليمات البرمجية في فصول أخرى، ولكن لن يتم تكرارها. الأنظمة الرئيسية هي:

- Form_Load يحتوي على جملة OpenDatabase.
- ShowNewRecord يعرض حقول بيانات فارغة.
- Picture1_Click تتحكم في التنقل حول مجموعة السجلات.

تعليمات برمجية ١٤-٨

```
' Movie Maintenance form
' Nick Snowdon  June 1998

Option Explicit

Private dbMovies As Database
Private rsMovies As Recordset
Private rsParticipants As Recordset
Private strMode As String

Private Sub Form_Load()
'  Connect to the database
   Set dbMovies = Workspaces(0).OpenDatabase("", False, False,
             "ODBC;DSN=movies;")
```

```
' Ensure the recordset is up to date
   RefreshRecordset
End Sub
Private Sub ShowNewRecord()
' Show a new blank record
   Dim iField As Integer
   For iField = 0 To 4
      txtMovie(iField) = ""
   Next              ' iField
End Sub
Private Sub ShowRecord()
' Show the current record
   Dim iField As Integer
   On Error GoTo ErrorHandler
' Take care of the situation if there is no current record
   If rsMovies.BOF And rsMovies.EOF Then
      ShowNewRecord
      Exit Sub
   ElseIf rsMovies.BOF Then
      rsMovies.MoveFirst
   ElseIf rsMovies.EOF Then
      rsMovies.MoveLast
   End If

' Display the fields
   For iField = 0 To 4
      txtMovie(iField) = "" & rsMovies(iField)
   Next ' iField
   Exit Sub
ErrorHandler:
   If Err.Number = 3146 Then
      MsgBox Errors(0).Description
   Else
      MsgBox Err.Description
```

```
    End If
End Sub
Private Sub lstRatings_Click()
    RefreshRecordset
End Sub
Private Sub Picture1_Click(Index As Integer)
'   Navigation controls
    Select Case Index
        Case 0              ' Top of recordset
            rsMovies.MoveFirst
        Case 1              ' Previous record
            If Not rsMovies.BOF Then
                rsMovies.MovePrevious
            End If
        Case 2              ' Next record
            If Not rsMovies.EOF Then
                rsMovies.MoveNext
            End If
        Case 3              ' Bottom of recordset
            rsMovies.MoveLast
    End Select

    ShowRecord
End Sub
```

تعرض التعليمات البرمجية ١٤-٩ التعليمات التالية المستخدمة في تنفيذ مجموعة السجلات القابلة للتحديث:

- RefreshRecordset والتي تعيد تكوين مجموعة السجلات بمعايير مختلفة إذا تم طلب ذلك. في الإصدار القابل للتحديث، يتم استدعائها مرة واحدة. مع ذلك لا يجب أن يتم ترميزها في حدث Form_Load، على الرغم من أنه من الممكن أن تريد تنشيطها باستخدام معايير إضافية عند نقطة معينة.
- cmdAction_Click والتي تتضمن طرق AddNew وEdit وUpdate وDelete التي يجري استخدامها لتغيير البيانات.

تعليمات برمجة ١٤-٩

```
Private Sub RefreshRecordset()
'  Ensure the recordset is up to date
   Dim strSQL As String

   strSQL = "SELECT movie_name, length, rating, year_of_release, " &
_           "movie_id FROM video_store.movies "
   strSQL = strSQL & " ORDER BY movie_id"
   Set rsMovies = dbMovies.OpenRecordset(strSQL, dbOpenDynaset)
   ShowRecord
End Sub

Private Sub cmdAction_Click(Index As Integer)
'  Execute the require action - insert, update or delete.
   Dim iField As Integer

   On Error GoTo ErrorHandler
   If Index = 0 Then          ' Add a new record
      rsMovies.AddNew
      ShowNewRecord
      strMode = "Add"

   ElseIf Index = 1 Then          ' Update the record
      If strMode <> "Add" Then
   '     You should not lock this if you are using transactions and
   '     then only for edits of existing records.
         dbMovies.Execute "SELECT * FROM movies WHERE movie_id =
" _
         & rsMovies(4) & " FOR UPDATE NOWAIT", dbSQLPassThrough

         rsMovies.Edit
      End If
       For iField = 0 To 2
```

```
            rsMovies(iField) = "" & txtMovie(iField)
        Next   ' iField
        rsMovies(3) = CInt(Val(txtMovie(3)))
        rsMovies(4) = CInt(Val(txtMovie(4)))
        rsMovies.Update
        strMode = ""
    ElseIf Index = 2 Then          ' Delete the record
        rsMovies.Delete
        ShowRecord
    End If

    Exit Sub
  ErrorHandler:
    If Err.Number = 3146 Then          ' ODBC Error
        If Errors(0).Number = 54 Then   ' Locked
            MsgBox "The record is currently locked: ORA-00054"
        Else
            MsgBox Errors(0).Description & ": " & Errors(0).Number
        End If
    Else                            ' Non-ODBC error
        MsgBox Err.Description
    End If
    rsMovies.CancelUpdate              ' Ensure update is canceled
    strMode = ""
End Sub
```

تعرض التعليمات البرمجية ١٤,١٠ التعليمات التالية المستخدمة في تنفيذ مجموعة سجلات للقراءة فقط:

- RefreshRecordset والتي تعيد تكوين مجموعة النتائج عند الطلب. في إصدار للقراءة فقط، هناك حاجة إلى استدعائها في كل مرة يتم فيها استخدام إجراء مخزون لتحديث جداول القاعدة.
- cmdAction_Click, والذي يتضمن استدعاءات الإجراء المخزون لإجراءات movies_insert2 و movies_update و movies_delete.

تعليمات برمجة ١٤-١٠

```
Private Sub RefreshRecordset()
'  Ensure the recordset is up to date
   Dim strSQL As String

   strSQL = "SELECT movie_name, length, rating, " & _
          "year_of_release, movie_id FROM video_store.movies "
   strSQL = strSQL & " ORDER BY movie_id"
   Set rsMovies = dbMovies.OpenRecordset(strSQL, dbOpenSnapshot)
   ShowRecord
End Sub
Private Sub cmdAction_Click(Index As Integer)
'  Execute the require action - insert, update or delete.
Dim strSQL As String

   On Error GoTo ErrorHandler

   If Index = 0 Then          ' Add a new record
      ShowNewRecord
      strMode = "Add"

   ElseIf Index = 1 Then          ' Update the record
      If strMode = "Add" Then
         strSQL = "BEGIN video_store.movies_insert2 (0,"
         strSQL = strSQL & "'" & txtMovie(0) & "', " ' Movie Name
         strSQL = strSQL & txtMovie(1) & ", "       ' Length
         strSQL = strSQL & "'" & txtMovie(2) & "', " ' Rating
         strSQL = strSQL & txtMovie(3)             ' Year
         strSQL = strSQL & "); END;"
         dbMovies.Execute strSQL, dbSQLPassThrough

      Else
```

```
        strSQL = "BEGIN video_store.movies_update ("
        strSQL = strSQL & rsMovies("movie_id") & ","
        strSQL = strSQL & "'" & txtMovie(0) & "', " ' Movie Name
        strSQL = strSQL & txtMovie(1) & ", "        ' Length
        strSQL = strSQL & "'" & txtMovie(2) & "', " ' Rating
        strSQL = strSQL & txtMovie(3)               ' Year
        strSQL = strSQL & "); END;"
        dbMovies.Execute strSQL, dbSQLPassThrough
      End If
      strMode = ""
      RefreshRecordset

    ElseIf Index = 2 Then          ' Delete the record
      dbMovies.Execute "BEGIN video_store.movies_delete (" & _
        rsMovies(4) & "); END;", dbSQLPassThrough
      RefreshRecordset
    End If

    Exit Sub

ErrorHandler:
    If Err.Number = 3146 Then          ' ODBC Error
      If Errors(0).Number = 54 Then   ' Locked
        MsgBox "The record is currently locked by another user : ORA-
00054"
      Else
        MsgBox Errors(0).Description & ": " & Errors(0).Number
      End If
    Else                           ' Non-ODBC error
      MsgBox Err.Description
    End If
End Sub
```

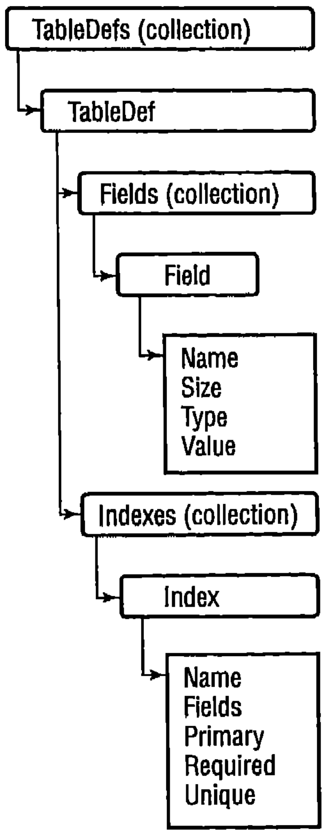
عرض هيكل قاعدة البيانات

هناك مرات عديدة سوف تريد فيها اكتشاف المزيد حول هيكل قاعدة بيانات أوراكل. على الرغم من إمكانية استخدام SQL*Plus للحصول على هذه التعليمات، إذا كان بإمكانك استعراضها من داخل فيجوال بيسيك، ولن تقوم هذه فقط بجعل التعليمات ملائمة لموردي قاعدة البيانات الآخرين ولكنها ستمكنك من أتمتة عمليات عديدة. سوف تضع هذا في موضع التطبيق في نهاية هذا القسم.

لبحث هيكل قاعدة البيانات، نحتاج أولا إلى بحث الوحدات والمجموعات والخصائص الموجودة في هيكل DAO. نتيجة لاحتواء ملفات تعليمات فيجوال بيسيك على عدد كبير من الخصائص، سوف نختار أكثر أهمية لمراجعتها هنا. يعرض الشكل ١٤,٤ الهياكل والخصائص المرتبطة.

الشكل ١٤-٤

هيكل مجموعة Tabledefs وأهم الخصائص.



الوصول إلى التجميعات

من الأهمية بمكان تفهم الطرق المتنوعة للوصول إلى التجميعات حيث أنها تشكل الأساس التي نعمل بها مع هياكل قاعدة البيانات. هناك ثلاث طرق للوصول إلى أي عنصر في التجميع:

- By ordinal number (بواسطة العدد الترتيبي) لفهرسة الوحدة مثل dbMovies.Tablesdefs(2).
- By object name "بواسطة اسم الوحدة"، مثل dbMovies.Tabledefs("movies").
- بواسطة التكرار ضمن التجميع باستخدام جملة FOR EACH، مثل:

For Each tbl In dbMovies.Tabledefs.

TableDefs

كما هو الحال بالنسبة لكل وحدة تقع تحت وحدة DBEngine والتي من الممكن أن تكون تجميع أو وحدة، TableDefs هي عبارة عن تجميع من وحدات TableDef. وحدة TableDef هي عبارة عن تمثيل لجداول أوراكل في قاعدة البيانات التي تم فتحها والوصول إليها. يتمن ملء التجميع بمعلومات عند التوصيل بقاعدة البيانات. نعم، يعتبر هذا الوصول الإضافي إلى قاعدة البيانات من المصروفات الإضافية، ولكنه يحدث بمعدل مرة واحدة عند كل اتصال. يعتبر هذا الموقف من الأمور الجيدة حيث أنه باستطاعة الوصول إلى قاعدة البيانات إضافة ثوان عديدة إلى زمن الاتصال.

لاكتشاف المزيد قدر الإمكان حول الجداول الواقعة تحت تصرفنا، يجب البحث داخل هذه الوحدة جيداً. الخاصية الوحيدة كثيرة الاستخدام بالنسبة لمستخدم أوراكل هي خاصية الاسم. يمكننا هذا من اكتشاف أسماء الجداول القابلة للوصول بالنسبة لاسم المستخدم. يمكننا استخدام هذه الخاصية كما هو معروض في تعليمات البرمجة ١٤-١١.

تعليمات برمجة ١٤-١١

```
Dim tbl As TableDef
   For Each tbl In dbMovies.TableDefs
      Debug.Print tbl.Name
   Next
```

ما الذي يقوم به هذا الرمز؟ أولاً إعلان متغير من نوع TableDef. ثم يقوم بالتكرار عبر كل TableDef في تجميع TableDefs ثم طباعة اسمه. إنها مواد سهلة في الحقيقة.

الحقول

تعتبر الحقول الموجودة في الجدول أكثر أهمية من أسماء الجداول. تحتوي كل وحدة Tabledef على تجميع حقول، والذي يحتوي بدوره على وحدة واحدة أو أكثر من وحدات الحقول. تم عرض الخصائص المهمة في وحدة الحقل في الجدول.١٤-١٠.

الجدول ١٤-١٠: خصائص وحدة الحقل التي تنطبق على أوراكل.

الخاصية	البيان
Name	اسم الحقل.
Size	حجم الحقل بالبايت.
SourceField	اسم الحقل من جدول مرفق (إذا كان متاحاً).
SourceTable	اسم الجدول المرفق (إن وجد).
Type	عدد يمثل نوع الحقل المترجم بواسطة JET. يحتاج هذا إلى الترجمة قبل كونه مفيداً في مقابل قاعدة البيانات.
Value	هي خاصية افتراضية ولكنها مناسبة عند استخدامها للإشارة إلى مجموعة سجلات بالنسبة لوحدة Tabledef تعتبر غير ذات معنى.

يمكنك تتبع هذه الأمور بسهولة تامة باستخدام رموز عينة في تعليمات ١٤-١٢.

تعليمات برمجة ١٤-١٢

```
Dim fld As Field
Dim tbl As TableDef
For Each tbl In dbMovies.TableDefs
   For Each fld In tbl.Fields
```

```
        Debug.Print fld.Name, fld.Type
    Next
Next
' Or ...
```

```
Debug.Print dbMovies.TableDefs("Movies").Fields("movie_name").Type
```

يمكنك القيام بالتنفيذ عند هذه النقطة وبدأ تطبيق صغير يعرض كل البيانات المرتبط بقاعدة البيانات. يمكنك حتى استخدام DAO لتبديل الجداول إذا كانت لديك الأذونات المطلوبة لتنفيذ ذلك. مع ذلك هناك أدوات أخرى مكتوبة بصورة أكثر احترافاً تقوم بتنفيذ هذه العملية بصورة أفضل، لذا ليس هناك معنى للتنفيذ ذلك بنفسك، حتى SQL*Plus من الممكن أن تعطيك طريقة عرض معقولة لقاعدة بياناتك.

ولذا من الممكن أن تسأل لماذا تريد استخدام هذه المعلومات على الإطلاق. حسناً، يمكننا على سبيل المثال استخدام المعلومات لتكوين وحدات سكريبت SQL تقوم بأكثر من إضافة حقل إلى جدول أو تكوين جدول جديد. سوف تشاهد مثال في قسم "تطبيق عملي من TableDefs" يأخذ إلى مستوى جديد.

الفهارس

تماماً كما يمكنك بحث حقول متنوعة الموجودة في جدول، يمكنك عرض الفهارس الموجودة به. من الممكن أن تحتوي كل وحدة TableDef على تجميع فهارس مرتبطة بوحدة فهرس واحدة أو أكثر. يجب فحص خاصية العد من تجميع الفهارس للتأكد من وجود فهرس بالفعل قبل محاولة الوصول إليه. من الواضح أنه ليس باستطاعتك كل المعلومات من جدول أوراكل التي من الممكن أن تريدها، ولكن يجب أن لا يكون هذا مفاجأة. يغطي الجدول ١٤-١١ الخصائص الهامة لمستخدمي أوراكل.

الجدول ١٤-١١: بعض الخصائص الهامة لوحدة الفهرس.

الخاصية	البيان
Fields	هذه الخاصية هي سلسلة تتضمن كل أسماء الحقول التي تكون الفهرس. كما تتوافر هذه الأسماء بصورة منفصلة في تجميع الحقول تحت وحدة الفهرس.

الجدول ١٤-١١: بعض الخصائص الهامة لوحدة الفهرس.

الخاصية	البيان
Foreign	تبين هذه الخاصية ما إذا كان الفهرس هو مفتاح ربط (غير متاح).
Name	تعطي هذه الخاصية اسماً للفهرس. (هذا هو اسم قيد أوراكل).
Primary	تبين هذه الخاصية ما إذا كان الفهرس مفتاح أساسي (غير متاح).
Unique	تبين هذه الخاصية ما إذا كان الفهرس مفتاح فريد. سيحتوي المفتاح الأساسي لأوراكل على خاصية فريدة معدة على صواب.

توجد وحدة حقل أسفل وحدة الفهرس، ولكنها لا تعتبر وحدة حقل صحيحة. حيث تحتوي على خاصية اسم متاحة فقط. للبحث عن خاصية نوع الوحدة، على سبيل المثال، يجب الرجوع إلى وحدة الحقل أسفل تجميع الحقول. تقوم الرموز التالية بتنفيذ ما يلي:

```
StrIndexName = tbl.Indexes(0).Fields(0).Name
Debug.Print tbl.Fields(strIndexName).Type
```

A Practical Application of TableDefs

تطبيق عملي TABLEDEF

كما وعدنا في الفصل الثامن، تحديث قاعدة البيانات، سوف نقوم الآن باستخدام المعلومات التي ناقشناها في الأقسام السابقة لتكوين برنامج أدوات مساعدة مفيد، والذي سيقوم بمسح كل الجداول في قاعدة البيانات وتكوين سكريبت حتى ثلاثة إجراءات مخزونة لكل جدول: واحداً لإضافة وواحداً لتحديث وواحداً لحذف صفوف البيانات. من الممكن أن يوفر هذا المنهج ساعات من العمل بالمقارنة بمنهج الترميز اليدوي.

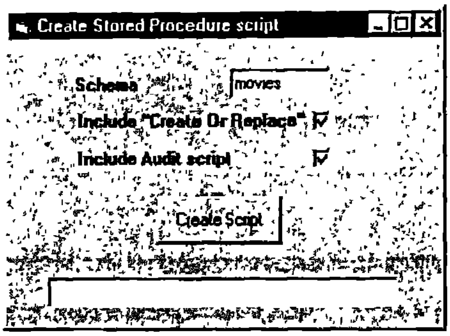
يجب معاملة يجب معاملة هذا الرمز كبداية للأفكار. على سبيل المثال، إجراءات الإضافة التي تقوم بتكوينها تتضمن معاملات خاصة بالمفاتيح الفريدة. إذا منت تستخدم المسلسلات (وهذا ما يجب أن تقوم به)، حينئذ من الممكن أن تريد تعديل المثال بحيث يتم إزالة الحقل المفتاح وبدلاً من ذلك إعداد القيمة من مسلسل مناسب.

تلميح إحدى الطريقتين التي أستخدمها أنا شخصياً إذا تم تمرير صفر لقيمة مفتاح، يقوم الإجراء بتكوين واحداً من المسلسل. بهذه الطريقة، يمكنك الحصول على أفضل ما في الطريقتين.

يوجد نموذج بسيط واحد فقط لتطبيق CreateProcs، وهذا معروض في الشكل ١٤-٥. كما يمكنك استعراضه، قم بإدخال اسم المخطط كمرشح بحيث لا تضطر إلى معالجة كل جدول مفود لديك حق الوصول إليه. يمكنك تحديد الخيار لتضمين صياغة Create أو Replace، والتي ستحتاجها إذا كنت ستقوم بتنفيذ سكريبت المخرجات من جلسة عمل SQL*Plus. في النهاية، يمكنك نقر خيار لتضمين وحدات سكريبت التدقيق في كل إجراء.

ملاحظة يتم عرض جزء واحد فقط من الرموز هنا. النسخة الكاملة معروضة على موقع Sybex على الويب على عنوان http://www.sybex.com.

الشكل ١٤-٥ صيغة CreateProcs



الحلقة الأساسية من البرنامج معروضة في إجراء CreateProcs انظر تعليمات البرمجة ١٤-١٣. حيث تقوم بالتكرار خلال داخل كل الجداول في تجميع TableDefs، وسوف تتضمن أي جدول لدى المستخدم حق الوصول إليه وليس فقط تلك الموجودة داخل المخطط الخاص بهم. لتقليل حجم المعالجة، يمكنك إدخال اسم المخطط في النموذج، إذا كان اسم الجدول متضمناً المخطط المدخل، حينئذ يتم معالجة الجدول.

تعليمات برمجة ١٤-١٣

```
Process each table in the collection
For Each tbl In dBTss.TableDefs
    iNumFields = tbl.Fields.Count   ' Number of fields in table
   If InStr(tbl.Name, UCase(txtSchema) & ".") And _
        iNumFields > 0 Then
      Find a unique index if one exists
     iIndexes = tbl.Indexes.Count
```

```
            Set fldUnique = Nothing
            If iIndexes > 0 Then
               For Each idx In tbl.Indexes
                  For Each fld In idx.Fields
                     Set fldUnique = fld
                     Exit For
                  Next
               Next
            End If
            Create the header
'           Create the SQL command
'           Create the audit
'           Create the footer
         Else
            TxtMessages.Text = "Ignoring table " & tbl.Name
         End If
         DoEvents
      Next tbl
```

أول شيء تقوم به حلقة المعالجة هو البحث عن مفتاح فريد حيث أن DAO لا تسمح باكتشاف قيد PRIMARY KEY. سوف يتم استخدام هذا المفتاح الفريد في تكوين إجراءات التحديث والحذف المخزونة. بمجرد العثور على المفتاح الفريد، تقوم الحلقة باستدعاء الإجراءات لتكوين عناوين ونصوص رسالة وتذييلات إجراءات مخزونة. في التعليمات البرمجية ١٤,١٣، يتم تمثيل هذه الاستدعاءات بواسطة تعليقات لتبسيط الرمز.

تم عرض الرمز الخاص بتكوين عنوان إجراء مخزون في التعليمات البرمجية ١٤-١٤. في هذا الإجراء تم تضمين المعاملات الرسمية لكل عمود موجود في الجدول فيما عدا اسم المستخدم والطابع الزمني. يتم التعامل مع إجراء الحذف المخزون بصورة مختلفة بعض الشيء حيث أنها تحتوي على معامل رسمي واحد فقط. سيتم استخدام المفتاح الفريد للتحديد الصف الذي سيتم حذفه.

تعليمات برمجة ١٤-١٤

```
' Create the header for a stored procedure.
Print #1, "PROCEDURE " & LCase(RemoveSchema(tbl.Name)) & _
        "_" & strType & " ("
```

```
' Write out one parameter for each field except
' user_name and time_stamp
iField = 0
For Each fld In tbl.Fields
   If fld.Name <> "USER_NAME" And fld.Name <> "TIME_STAMP"
Then
      iField = iField + 1
      If iField > 1 Then
         Print #1, "   ,p_" & LCase(fld.Name) & _
            " " & FieldType(fld.Type)
      Else
         Print #1, "    p_" & LCase(fld.Name) & _
            " " & FieldType(fld.Type)
      End If
   End If
Next fld
Print #1, "  ) IS "
```

لكتابة المعاملات الرسمية، يجب معرفة نوع البيانات لكل عمود. سوف يسبب هذا صعوبة صغيرة حيث أن أنواع ODBC لا تحتوي على تحويل رأس برأس إلى أنواع بيانات أوراكل. تقوم الرموز الموجودة في التعليمات البرمجية ١٤,١٥ بعمل هذا التحويل نيابة عنا.

تعليمات برمجة ١٤-١٥

```
Function FieldType(intType As Integer) As String
'  Convert the field type to an Oracle column type
   Select Case intType

      Case dbByte
         FieldType = "NUMBER"
      Case dbInteger
         FieldType = "NUMBER"
```

```
        Case dbLong
            FieldType = "NUMBER"
        Case dbSingle
            FieldType = "NUMBER"
        Case dbDouble
            FieldType = "NUMBER"
        Case dbDate
            FieldType = "DATE"
        Case dbText
            FieldType = "VARCHAR2"
        Case dbLongBinary
            FieldType = "RAW"
        Case dbMemo
            FieldType = "VARCHAR2"
        Case Else
            FieldType = "Unknown"
    End Select
End Function
```

في النهاية، يتم عرض النتائج التي انتظرتها في التعليمات البرمجية ١٤,١٦. المخرجات هي إجراء PL/SQL يمكنك وضعها بصورة مباشرة داخل.باني إجراءات أوراكل. في الواقع، إذا كنت تقوم بتكوين إجراءات كثيرة، يحتوي التطبيق على مربع اختيار خاص بإضافة جملة CREATE أو REPLACE في مقدمة كل إجراء واتباعه بشرطة مائلة للأمام. بهذه الطريقة، يمكنك تنفيذ ملف المخرجات بالكامل في جلسة عمل SQL*Plus.

ملاحظة إذا كان لديك جداول ذات أعمدة بيانات أولية، لن تكون قادرا على استخدام DAO لاستردادها. سيكون عليك استخدام واحدة من تطبيقات الوصول إلى البيانات الأخرى.

تعليمات برمجة ١٤-١٦

```
PROCEDURE customers_delete
 ( p_customer_id NUMBER
 ) IS
/* This procedure deletes a row in the NICK.CUSTOMERS table */
BEGIN
 DELETE FROM NICK.CUSTOMERS
  WHERE customer_id = p_customer_id;

/* Add the procedure audit trail */
 INSERT INTO procedure_audit VALUES (
  audit_sequence.NEXTVAL,
  'customers_delete',
  p_customer_id
  ,SYSDATE, USER);
 COMMIT;
EXCEPTION
/* Whatever the problem just rollback and pass the error back to Visual
Basic */
 WHEN OTHERS THEN
  ROLLBACK;
  RAISE;
END customers_delete;
```

حتى لو لم تكن تخطط لاستخدام هذه الطريقة بالضبط، فهي فكرة تستحق الوضع في الاعتبار، حيث أن هناك عدد لا يحصى من الأفكار المشابهة من الممكن أن تجعل العمل أسهل.

مردودات وحدات الوصول إلى البيانات

تعتبر DAO من التقنيات المتقادمة. يجب تقبل أي شيء أصبح أقل أهمية مع DAO. فما زالت تستخدم مع عدد ضخم من التطبيقات، ولكن هناك بالتأكيد عيوباً بالمقارنة بتقنيات أكثر حداثة.

تحتوي DAO على نموذج وحدات ضعيفة بمستوى أقل من الفعالية من كل تقنيات الوصول إلى البيانات.

- تم تصميم DAO لـ JET ولم يتم موالفتها لمصادر بيانات ODBC.
- لا يمكنك استرداد معاملات المخرجات من إجراءات مخزونة.
- DAO ليست بالتقنية الموائمة لمدى واسع من أنواع مصادر البيانات.
- لقد وصلت هذه التقنية إلى نهاية الطريق ولن يتم تحديثها بعد ذلك.

خُلاصة

لقد قام هذا الفصل بتغطية كلل الميزات الرئيسية لاستخدام DAO مع أوراكل. بما في ذلك تكوين اتصالات قاعدة البيانات واسترداد مجموعات السجلات للقراءة فقط والقابلة للتحديث وتنفيذ جمل SQL واستدعاء الإجراءات المخزونة. ولقد ناقشنا أمثلة عملية لطريقتين رئيسيتين لاستخدام DAO لعرض وتحديث البيانات. ولقد انتهينا بعرض إجمالي لهيكل قاعدة البيانات واستخدام هذه المعلومات لإنتاج سكريبت لتكوين إجراءات مخزونة.

على الرغم من الوظائف التي يمكنك استخراجها من DAO، فهي نموذج وحدة محدود وصناعي إلى حد ما عند تطبيقه على قاعدة بيانات أوراكل. سوف ننتقل إلى بقية تقنيات الوصول إلى البيانات لتوفير وظيفة أفضل كما سنقترب أكثر وأكثر من الطريقة التي يعمل بها أوراكل فعلاً.

الجزء الرابع

أساليب متقدمة لأوراكل

- وحدات البيانات البعيدة
- ODBCDirect
- وحدات بيانات ActiveX
- وحدات أوراكل التي تخص OLE

١٥

وحدات البيانات البعيدة

تعرفنا حتى الآن على كيفية استخدام Data Access Objects (DAO) للوصول إلى قاعدة البيانات لأوراكل. وكما رأيت فإن DAO لا يعتبر نموذج مثالي لأنه لا يعطي الوصول إلى الفعالية التامة لأوراكل. ومع هذا فإن هذه الحقيقة لا تأخذ أي من DAO لأنه في الإصدار الجديد (3.51 وODBCDirect) يظهر أكثر سرعة. ولقد تم كتابة تطبيقات متعددة بواسطة استخدامه ومع هذا لم يتم تصميم DAO حقيقة خصيصاً لقواعد البيانات البعيدة الذكية ولكنه مصمم لمشغل قواعد البيانات JET ومن هناك بدأ في التطور.

وفي أدنى تنفيذ له، يتم تنفيذ ODBC كواجهة برمجة للتطبيق "Application Programming Interface" لجعل ODBC أكثر إتاحة لمعظم المبرمجين. ولقد قامت مايكروسوفت بتزويد نموذج DAO الذي أسدل الستار على تعقيد API ولكن على حساب الفعالية. ولقد أتت (RDO) Remote Data Objects 1.0 في VB 4.0 كغلاف على ODBC API للاهتمام بأي تفاصيل دقيقة ولفتح طاقة المشغل ODBC للمطور. ثم مع VB 5.0، أتى RDO 2.0 الذي جلب جداول زيادة إضافية وبيئة غنية. يحصل RDO على فعالية أكثر من بنية ODBC على قدر توقعك وعليك ملاحظة أن RDO يعمل في بيئة ٣٢بيت وأن له إصدار أدنى للفيجوال بيسك ٦.

وعليك الانتباه إلى نقطة أن RDO متاحة فقط مع Enterprise Edition للفيجوال بيسك، لذا توجد هناك التكلفة المرخصة الإضافية المتعلقة. يمكنك استخدام RDO فقط في بيئات التطور، مثل VB و++VC، ولن يوجد لديك هذا الخيار إذا كنت مطور لمايكروسوفت أوفيس. وفي هذه الحالة، عليك استخدام أسلوب مختلف مثل الذي سوف نقوم بمناقشته في الفصل ١٦ ODBCDirect و١٧ وحدات بيانات ActiveX.

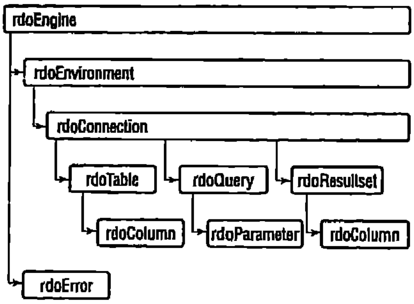
أن حقل القراءة يعتبر ومع متعدد، سوف نركز على الموضوعات التي سوف تكون بحاجة إليها لإنتاج تطبيقات مقبولة مع أوراكل. والموضوعات التي سوف نقوم بتغطيتها هي:

- هيكل وحدات RDO.
- قواعد البيانات والوصلات.
- العمل مع مجموعات النتائج rdo.
- تنفيذ أوامر SQL.
- استخدام وحدات rdoQuery.
- استدعاء الإجراءات المخزنة.
- التشغيل غير المتزامن.
- استرداد جداول PL/SQL.
- معالجة الأخطاء.

- عيوب RDO.
- مستقبل RDO.

هيكل الوحدة

يعتبر هيكل الوحدة RDO مختلف عن الهيكل الذي شاهدناه في الفصل الرابع عشر، وحدات الوصول للبيانات بشكل ملحوظ. ويوجد هنا تعقيد أكثر بشكل واضح وبعض من المصطلحات قد تتغير كما هو موضح في الشكل ١٥-١. ويوجد سبب كافي لهذا: فهذه المصطلحات تشبه مصطلحات ODBC وAPI ونموذج قاعدة البيانات العلاقي الذي تم تصميم ODBC لكي تتواجه معه. ومع هذا فمن وجهة نظر المطور، يؤدي هذا إلى للبس أكثر. ويعد هذا سوء حظ لأنه مع وحدات البيانات Active X، انصرفنا الآن من النظر إلى قواعد البيانات العلاقية كأساس لكل الأشياء ورجعنا ثانية إلى المصطلح Recordset. ولحسن الحظ يحصل المطورين على تدريب منتظم في التعامل مع الأسماء المتغيرة للوحدات المعتادة.



الشكل ١٥-١
هيكل وحدات RDO.

rdoEngine

تعتبر وحدة rdoEngine وحدة مستوى القمة في هيكل RDO. يوجد rdoEngine وقد يحتوي على واحدة أو أكثر من rdoEnvironments في مجموعة. ويتم استخدام rdoEngine لتحديد rdoEnvironments التي نريد تكوينها؛ ومع هذا لا يمكنك تكوين rdoEngine جديد. يحتوي rdoEngine على مجموعات rdoEnvironments وrdoErrors.

rdoEnvironments

تعتبر وحدة rdoEnvironments تجميع من واحدة أو أكثر من وصلات قاعدة البيانات البعيدة، وتتبع كل هذه الوصلات نفس مجال الحركات. يعني هذا أنك إذا كنت تريد استخدام التحكم في الحركات عن طريق RDO (بمعنى أنك تستخدم BeginTrans وCommitTrans وRollbackTrans) ثم تقوم بتثبيت التغييرات ثم بعد ذلك يتم تثبيت كل التغييرات في نفس البيئة

ولكن على وصلات مختلفة في نفس الوقت. وتكون درجة الفعالية نفس DAO Workspace، ولكن يكون لك تحكم أكبر فيها. وكما هو الحال مع DAO، يمكنك أن تقبل البيئة الافتراضية في معظم الأحيان، rdoEnvironment(0) التي لن يتم تكوينها ما لم تتم الإشارة إليها أولا.

rdoConnections

تمثل وحدة rdoConnections وصلة إلى قاعدة بيانات بعيدة معينة، مثل قاعدة البيانات لأوراكل الخاصة بك. تحتوي rdoConnection على تجمعات rdoQueries وrdoResultsets وrdoTables. وتعتبر الوصلات المستوى الأول للهيكل الذي يتطلب منك بالفعل البدء في عمل أي شئ، ويمكنك أن تقبل البيئة الافتراضية وأن تتجاهل rdoEngine، ولكن لابد أن تعطي اهتمام أكثر للوصلات. وسوف نغطي هذه الأشياء في القسم التالي. لاحظ أنه يمكنك أيضا استخدام أساليب التحكم في الحركات عن طريق وحدة rdoConnection وهذه من المحتمل أنسب طريقة لاستخدامهم.

rdoResultsets

تعتبر rdoResultsets مجموعة من الصفوف تم استرجاعها من استعلام وهي تساوي Recordset في DAO. وسوف ننظر إلى استخدام rdoResultsets بتعمق أكثر فيما بعد بهذا الفصل. ويمكن أن تحتجز rdoResultset واحدة أكثر من مجموعة مرتبطة من الصفوف. وتعتبر معالجة مجموعات النتائج المتعددة في وحدة rdoResultset ميزة يتم تصميمها للاستخدام مع خادم SQL ولكن يمكن تصميمها للعمل مع أوراكل كما سوف تشاهد في القسم "استرجاع جداول PL/SQL". وسوف نتركها الآن للفصل السابع عشر وحدات بيانات ActiveX، لتحويل هذه الطريقة إلى حيز التطبيق. وفي الغالب، سوف يكون لديك مجموعة نتائج منفردة من أوراكل في وحدة rdoResultset الخاصة بك.

rdoQueries

تعتبر rdoQueries وحدة تقوم بتنفيذ غرضين أساسين: الأول: فهي طريق إلى تجهيز استعلام SELECT بواسطة المعاملات بحيث لا تكون في حاجة إلى تكوينه ثانية في كل مرة تقوم فيها بتغيير واحدة أو أكثر من قيم المعامل. وإذا قمت باستخدامه لمرات عديدة، فهذه طريقة لتجنب التحليل المتكرر للجملة التي تقوم بها ODBC في كل مرة تقوم بتشغيله. واستخدام rdoQueries الثاني الرئيسي هو استدعاء الإجراءات المخزنة واسترجاع معاملات المخرجات.

rdoTables

يعتبر rdoTables تمثيل للجدول الأساس في قاعدة البيانات لأوراكل. يوجد rdoTable واحد في مجموعة rdoTables في كل جدول لأوراكل يتم لك الوصول إليه (قد تندهش كثيرا عند معرفة الجداول التي يقوم بتغطيتها إذا كان لديك امتيازات المطور). ويعد هذا قابل للمقارنة مع

وحدة TableDef في DAO. تتيح لك بعض قواعد البيانات استخدام rdoTable كأساس للاستعلام ولكن لا ينطبق هذا على أوراكل. مع أن DAO تتيح لك الإضافة والتعديل والحذف، فإن RDO لا توجد بها هذه الإمكانية؛ ومع هذا لا تعتبر هذه هي الطريقة المفضلة لتعديل هيكل قاعدة البيانات بأي طريقة.

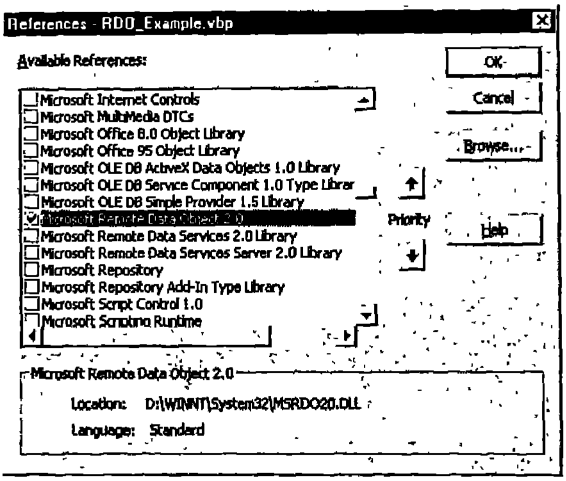
قواعد البيانات والوصلات

للاستفادة من أي خصائص لوحدات البيانات البعيدة، علينا أولاً إعداد مرجع في التطبيق الخاص بنا إلى مكتبة RDO في المشروع الذي بين أيدينا. ويمكن لنا إجراء هذه العملية بواسطة الخطوات التالية:

١. اختر References „ Project لفتح مربع الحوار References كما هو موضح في الشكل ١٥-٢. وسوف تشاهد قائمة بكل مكتبات الوحدة التي يمكنك الإشارة إليها في التطبيق الخاص بك.

٢. راجع الخيار المسمى Microsoft Remote Data Object 2.0.

٣. انقر OK.



الشكل ١٥-٢

مربع حوار References مع مكتبة RDO التي تم لها الفحص

توجد طريقتان لتكوين وصلة إلى قاعدة البيانات: طريقة قصيرة وطريقة مطولة. لاتباع التوجيه القصير، عليك القفز مباشرة إلى القسم "تكوين وصلة". وتقوم الطريقة المطولة بإعطاء نفس النتيجة التي تتيح لك تحكم أكثر في البيئة التي تقوم بتكوينها.

وإذا قبلت البيئة الافتراضية، يمكنك تجاهل وحدة rdoEngine بسعادة. ومع هذا إذا كنت تريد تحكم بالكامل في أي وحدات rdoEnvironment تقوم بناءا على ذلك بتكوينها، يمكنك استخدام الخصائص لوضع القيم الافتراضية. يوضح الجدول ١٥-١ الافتراضات التي يمكنك وضعها. وعند العمل مع أوراكل، سوف تجد أن مشغل المؤشر ClientBatch هو الخيار الأفضل لـ rdoResultsets القابلة للتحديث لأنها تقوم بالتنفيذ بشكل أفضل كثيرا من المؤشرات الأخرى. وسوف تكون بحاجة إلى وضع مشغل المؤشر هذا لأنه لا يعتبر الافتراضي،وخصائص rdoEnvironment الأخرى ذات استخدام محدود.

الجدول ١٥-١: خصائص rdoEngine التي يمكنك استخدامها لإعداد Default rdoEnvironment Features.

الخاصية	البيان
RdoDefaultCursor	هذه الخاصية هي مشغل المؤشر الافتراضي. والقيم الممكنة التي يمكنك تحديدها موضحة في الجدول ١٥-٢
RdoDefaultUser	تعتبر هذه الخاصية كلمة المرور المتعلقة بـ RdoDefaultUser
RdoDefaultErrorThreshold	تعتبر هذه الخاصية المرشح لرموز الأخطاء الجديرة بالاعتبار. وتعتبر هذه الخاصية غير هامة في بيئة أوراكل.
RdoDefaultLoginTimeout	تعتبر هذه الخاصية وقت الحد الأقصى المسموح به في إجراء محاولة عمل وصلة مع قاعدة البيانات.
RdoVersion	تشير هذه الخاصية إلى رقم الإصدار لمكتبة RDO.

الجدول ١٥-٢: خيارات المكتبة المتاحة لمشغل المؤشر الافتراضي في وحدة rdoEngine.

الثابت	القيمة	البيان
RdUseIfNeeded	صفر	يعتبر هذا الثابت القيمة الافتراضية. يقوم ODBC Driver Manager بتحديد مكتبة مشغل المؤشر للاستخدام. ومع أوراكل، سوف يقوم بتحديد مكتبة المؤشر ODBC (rdUseOdbc)

الجدول ١٥-٢: خيارات المكتبة المتاحة لمشغل المؤشر الافتراضي في وحدة rdoEngine.

الثابت	القيمة	البيان
RdUseOdbc	١	يستخدم هذا الثابت مكتبة مؤشر ODBC على ماكينة المزود.
RdUseServer	٢	يستخدم هذا الثابت المؤشرات الجانبية للخادم. وهذا الثابت غير متاح لأوراكل.
RdUseClientBatch	٣	يستخدم هذا الثابت مكتبة مؤشر ClientBatch التي تم تطويرها من أجل FoxPro وأسرع بشكل هام من خيار rdUserOdbc. وعليك استخدام هذا مع rdoResultsets القابلة للتحديث.
RdUseNone	٤	يدل هذا الثابت على أن المؤشر سوف يكون للإرسال فقط وللقراءة فقط وصف واحد لكل مرة للمؤشر. ولا يعتبر هذا الثابت مؤشر حقيقة على الإطلاق ولكن يقوم بتطبيق الطريقة التي يقوم فيها أوراكل بإنتاج مجموعة النتائج وجعلها في غاية الكفاءة.

تذكر أن الخيارات الموضحة في الجدول ١٥-٢ خاصة بتحديد مكتبة مشغل المؤشر وليس المؤشر ذاته. ومع هذا يوجد لدينا، مثل مطوري أوراكل، إمكانية اختيار محدودة وبسيطة. وإذا كنت متأكد أنك لن تكون بحاجة إلى أي مجموعة نتائج قابلة للتحديث في التطبيق الخاص بك، فعليك اختيار rdUseNone وإلا عليك تحديد RdUseClientBatch الذي يعد بشكل ملحوظ أسرع من rdUseOdbc.

يوضح المثال في التعليمات البرمجية ١٥-١ كيفية استخدام خصائص rdoEngine لإعداد الوصلات التي يقوم بعملها نتيجة لذلك. وفي هذه الحالة نقوم بإعداد معلومات المستخدم الافتراضية. وبالافتراض أنها صحيحة، فلن يكون هناك أخطاء أو الحاجة إلى Driver Manager للمطالبة بمعلومات. وقد تأتي هذه المعلومات من مربع حوار الدخول الخاص بك. وفي الحقيقة، قد تقوم بتحديد معلومات المستخدم عندما تقوم بتكوين الوصلة، مع أن هذا يعد المكان المناسب لوضع القيم إذا كنت تنوي الاتصال بطبعات قواعد بيانات متعددة في نفس

الوقت. وتقوم تعليمات البرمجة أيضا بتحديد مكتبة مشغل المؤشر ومدة الوقت المحدد لعمل الوصلة.

تعليمات برمجة ١٥-١

```
With rdoEngine
    .rdoDefaultCursorDriver = rdUseClientBatch
    .rdoDefaultUser = "video_user"
    .rdoDefaultPassword = "ummagumma"
    .rdoDefaultLoginTimeout = 5
End With
```

تكوين وحدة rdoEnvironment

قد لا تكون بحاجة إلى تكوين وحدة rdoEnvironment لأن الوحدة الافتراضية تكون دائما كافية، ومع هذا إذا كنت تريد استخدام التحكم في الحركة الخاص بقواعد البيانات المختلفة أو إذا كنت تريد أن يكون لديك وصلات لمستخدمين مختلفين، فيجب أن تكون قادر على تكوين البيئات الخاصة بك.

ولتكوين وحدة rdoEnvironment، استخدم الصيغة التالية:

Set env_variable = rdoCreateEnvironment (name, user, password)

وتحتوي الوحدة على المعاملات التالية:

- **name**: سلسلة تقوم بتعريف وحدة rdoEnvironment بشكل فريد.
- **user**: سلسلة تقوم بتعريف مالك وحدة rdoEnvironment الجديدة.
- **password**: سلسلة مكونة من ١٤ حرف طول وتحتوي على كلمة المرور للدخول إلى وحدة rdoEnvironment الجديدة.

تعتبر طريقة rdoCreateEnvironment أقل الطرق أهمية ومن الممكن جداً أن تغفلها. وعلى كلٍ، توضح التعليمات البرمجية ١٥-٢ مثال حول إعداد البيئة. وتصبح معاملات User وPassword القيم الافتراضية لهذه البيئة في حالات عدم التزويد بسلسلة اتصال. وبعد تنفيذ هذا، سوف يكوم لديك بيئتان (متضمنتان البيئة الافتراضية) دخول إلى أكثر من معرف مستخدم واحد ولكن مع إمكانية مجالات مختلفة للحركة في كل بيئة.

تعليمات برمجة ١٥-٢

```
Dim en As rdoEnvironment
Set en = rdoCreateEnvironment("Env2", "JOE", "saucerful")
```

تكوين وصلة

أساس العمل مع RDO هو الاتصال بقاعدة البيانات لأوراكل الخاصة بك. وتكوين وحدة rdoConnection الشيء الأول الذي عليك القيام به قبل البدء في أي شئ آخر. وبعكس محرك JET تحت DAO، فلدينا تحكم الآن في الوصلات ومن الممكن الاستفادة بشكل أفضل من الموارد المتاحة لنا. ولا يعتبر هذا مختلف كثيرا عن فتح قاعدة البيانات في DAO ما عدا أنه يوجد نموذج مختلف قليلا. والشيء الرئيسي الذي يدعوا للقلق هو سلسلة الاتصال لقاعدة البيانات.

ولأن RDO يعتبر نظام على أساس الوحدة، فعندما نقوم بفتح الوصلة بقاعدة بيانات عن بعد، فإننا نقوم بالفعل بتكوين وحدة تمثل هذه الوصلة. والطريقة الطبيعية لتكوين وحدة الوصل هي أولا التصريح بوحدة من النوع الصحيح ثم استخدام طريقة OpenConnection لتكوين الوحدة. وصيغة OpenConnection كالآتي:

```
Set connection = environment.OpenConnection(dsName, prompt,
readonly, connect, options)
```

والمعاملات المختلفة لطريقة OpenConnection كالآتي:

DsName: استخدم هذا المعامل لتزويد Data Source Name (DSN) (اسم مصدر البيانات) للوصلة. وسوف تمر على DSN هذا في سلسلة الوصلات وإذا كنت تريد إجراء ذلك، عليك وضع سلسلة بطول صفر ("") مثل حقل dsName. والميزة الرئيسية لتمرير قيمة dsName هي أنه يمكنك تعريف الوصلة به.

Prompt: يتحكم هذا المعامل في كيفية عرض مربع حوار الدخول ODBC. والقيم الممكنة الموضحة في الجدول ١٥-٣. وعليك أن تهدف إلى تحديد معامل التوجيه الذي يسبب للمستخدم أقل كمية من اللبس مع الإتاحة لهم بالاتصال بأي قاعدة بيانات هم بحاجة إليها، وإذا كان من الممكن، يجب تجنب استخدام مربع حوار الدخول ODBC كلية.

Readonly: عليك إعداد هذا المعامل على False إذا كنت تنوي استخدام مجموعات النتائج القابلة للتحديث في هذه الوصلة، وإلا عليك وضعها على True

Connect: يقوم هذا المعامل باحتجاز سلسلة اتصال ODBC التي شاهدناها في الفصل ١١، تقنية جانب المزود. وإذا قمت بإعطاء قيمة في السلسلة لمعامل dsName، فعليك إذا تحديد حقل DSN في سلسلة الاتصال.

Options: الخيار الوحيد الهام لمستخدمي أوراكل هو ثابت rdAsynchEnable الذي يعني أن الوصلة سوف تكون غير متزامنة. وبمجرد بدء عملية الاتصال هذه، عليك العودة إلى التحكم في التطبيق الخاص بك لاستمرار لمعالجة. وسوف تكون بحاجة إلى مراجعة تمام النجاح لعملية الاتصال قبل استخدام الوصلة (على سبيل المثال، يمكنك استخدام حدث Connect). ولأنه سوف يتم لك الاتصال بشكل طبيعي عند البدء وأن الوصلات لن تستغرق وقت طويل، فهذا الخيار لا يعتبر هام مثلما تقوم بتكوين rdoResultsets غير متزامنة.

الجدول ١٥-٣: القيم المتاحة لمعامل التوجيه.

الثابت	القيمة	البيان
RdDriverPro.npt	صفر	يعرض ODBC Driver Manager مربع حوار Data Sources ويقوم بعمل سلسلة وصل من البيانات التي تم تزويدها.
RdDriverNoPrompt	١	يقوم ODBC Driver Manager باستخدام سلسلة الوصل التي قمت بتزويدها في dsName والاتصال بالبيانات. وإذا لم تقم بتزويد معلومات كافية، فسوف تحصل على أخطاء متكررة.
RdDriverComplete (افتراضي)	٢	إذا كان معامل الوصل يتضمن DSN، فسوف يتم استخدامه للاتصال، وإلا سوف يتم عرض مربع حوار Data Sources.
RdDriverComplete Required	٣	هذا الثابت نفس rdDriverComplete إلا أنه لابد من طلب معلومات ليتم الدخول.

تلميح كما تم الذكر في الفصل الثالث عشر، موضوعات عن التصميم باستخدام فيجوال بيسك، لا يعتبر السماح لأي مستخدم مشاهدة أي من مربعات حوار الدخول إلى ODBC Driver Manager الحل الأفضل. ولكن عليك القيام بتكوين مربع حوار الدخول الخاص بك باسم مستخدم وكلمة مرور واستخدام الثابت dDriverNoPrompt للاتصال. إذا كانت خدمات البيانات الخاصة بك على الطبقة الوسطى، فيعتبر هذا النوع من التقريب إلزامي.

توضح التعليمات البرمجية ١٥-٣ الإجراء ConnectToDatabase الذي يتصـــل بقــاعدة البيانات من خلال البيئة الافتراضية. وفي هذه الحالة، نكون قد قمنا بتزويد كل معاملات الدخـول المطلوبة، ولذلك سوف يتم استخدام rdDriverNoPrompt. وسوف يسبب هذا فشـــل Driver Manager إذا كانت المعاملات خاطئة وكانت هذه المعاملات أفضل من عرض مربعات الحـوار Driver Manager أو Driver.

تعليمات برمجة ١٥-٣

```
Dim cn As rdoConnection
Private Sub ConnectToDatabase ()
'  Declare and open a connection to the database
   Dim strConnection As String
    StrConnection = "DSN=movies; UID=video_user;
PWD=ummagumma;"
   Set cn = rdoEngine.rdoEnvironments(0).OpenConnection _
      ("movies", rdDriverNoPrompt, False, strConnection)
End Sub
```

تكوين وصلات مستقلة بذاتها

وكطريقة بديلة لتكوين وصلة هيَ اســـتخدام وصلـــة مســـتقلة بذاتـــها. عـــن طريـــق تكوين وحدة وصل بواسطة تعريفه بـ New كمـــا يلـــي:

```
Set cn as New rdoConnection
```

والآن يوجد لديك وحدة يمكنـــك معالجتـــها بواسـطة الخصــائص مثـــل Connect، حتـــى تكـــون جـــاهز للاتصـــال فعليـــا بقـــاعدة البيانـــات باســـتخدام طريقـــة EstablishConnection.

وصلات DSN-less

الطريقة التقليدية لعمل اتصال بمصدر البيانات ODBC هي من خـــلال Data Source Name المسجل. أولا عليك إعداد DSN على كل ماكينة مزود تقوم بتشغيل التطبيق الخاص بك. وبعد أن يتم تشغيل التطبيق الخاص بك، على التطبيق أن يبحث عن DSN في Registry والكشف عـــن مشغل ODBC الذي سوف يتم استخدامه وسوف يستغرق هذا وقت قصير.

ومع RDO 2.0 يمكنك تكوين وصلة بدون DSN، والتي تسمى وصلة DSN-less. ولإجراء هذا، عليك تحديد اسم مشغل ODBC في سلسلة الوصلة بعد الكلمة الأساسية DRIVER. وعليك التأكد من تحديد الاسم الدقيق للمشغل (الذي يمكنك الكشف عنه من برنامج ODBC Admin) وتقوم بتضمينه في أقواس صغيرة. وكمثال، بافتراض أنك تريد الاتصال بقاعدة البيانات MOVIE بأكفأ طريقة ممكنة توضح التعليمات البرمجية ١٥-٤ البديل لإجراء ConnectToDatabase الذي يقوم بهذا من خلال وصلة DSN-less. وفي هذا المثال، سوف نستخدم أيضا مكتبة الخلية ClientBatch.

تعليمات برمجية ١٥-٤

```
Dim cn As rdoConnection
Private Sub ConnectToDatabase()
Dim strConnection As String
With rdoEngine
.rdoDefaultCursorDriver = rdUseClientBatch
.rdoDefaultUser = "video_user"
.rdoDefaultPassword = "ummagumma"
rdoDefaultLoginTimeout = 1
End With
StrConnection = "SERVER=movies;" & _
"DRIVER={Microsoft ODBC for Oracle};"
Set cn = rdoEngine.rdoEnvironments(0).OpenConnection _
("", rdDriverNoPrompt, False, strConnection)
End Sub
```

وعليك أن تلاحظ في التعليمات البرمجية ١٥-٤ أنه يوجد لدينا كلمة مرور واسم مستخدم افتراضي ومحدد في مستوى rdoEngine. لذا ليس مطلوب التكرار في سلسلة الوصلة وعلينا أيضاً تحديد معامل DSN في سلسلة الوصلة كسلسلة طولية بقدر صفر، ومع هذا علينا تحديد بندين جديدين:

Driver: الاسم الدقيق لمشغل ODBC الذي تريد استخدامه، كما هو موضح في برنامج إدارة ODBC. لا يستطيع معرفة حالة الأحرف ولابد من تضمينه بين أقواس.

Server: اسم Oracle Service الذي تريد الاتصال به أعني الاسم الذي يظهر في tnsnames.ora.

وكما ذكرنا سابقا، تعتبر وصلة DSN-less طريقة جيدة للاتصال بقاعدة البيانات، على الرغم أن رسائل الخطأ التي تتلقاها من الممكن أن تكون أكثر من شفرة صغيرة. وفي حالة تلقي أخطاء، عليك إجراء المراجعة مرتين على أسماء الخادم والمشغل.

خصائص rdoConnection

توجد كمية لا بأس بها من الخصائص بوحدة rdoConnection، وهذه الخصائص موضحة في الجدول ١٥-٤ ومعظم هذه الخصائص متعلق بالمعلومات فقط ومن المحتمل أن يتم استخدامه بشكل كبير. ويجد خاصتان جديرتان بالإشارة إليهما. يمكنك استخدام StillExecuting لمشاهدة ما إذا كان يتم تنفيذ الاستعلام الغير متزامن في الوصلة من عدمه. (وأفضل طريقة لإجراء هذا هي بواسطة استخدام حدث Connect، الذي سوف نقوم بتغطيته في القسم التالي). والخاصية الهامة الأخرى هي خاصية QueryTimeout التي تتيح لك تجنب أي استعلامات تزيد عن الحصة المحددة لموارد الخادم.

الجدول ١٥-٤: خصائص rdoConnection.

الخاصية	البيان
AsyncCheckInterval	توضح هذه الخاصية الرقم بأجزاء الألف من الثانية بين كل بحث وآخر لمشاهدة ما إذا كان التشغيل غير المتزامن كامل من عدمه (الافتراضي ١٠٠٠ م.س).
Connect	توضح هذه الخاصية سلسلة الوصلة المستخدمة في عمل الوصلة.
CursorDriver	توضح هذه الخاصية الرقم الذي يشير إلى مشغل الخلية المستخدم.
LoginTimeout	توضح هذه الخاصية المدة المستغرقة (بالثواني) في عمل الوصلة.
LogMessages	تمكن هذه الخاصية ODBC من تتبع الدخول.
Name	توضح هذه الخاصية اسم الوصلة.
QueryTimeout	توضح هذه الخاصية المدة المستغرقة (بالثواني) للاستعلام عن هذه الوصلة.

الجدول ١٥-٤ : خصائص rdoConnection.

الخاصية	البيان
LastQueryResults	تشير هذه الخاصية إلى آخر rdoResultset.
RowsAffected	توضح هذه الخاصية عدد صفوف البيانات المتأثر بآخر طريقة Execute. ولا يقوم أوراكل بإرجاع البيانات.
StillConnecting	تدل هذه الخاصية على أن طلب الوصلة غير المتزامن ما زال جاري تحت الحركة.
StillExecuting	تدل هذه الخاصية على أن طريقة Execute ما زالت تحت الحركة.
Transactions	تدل هذه الخاصية على ما إذا كانت الوصلة تدعم الحركات من عدمه.
UpdateOperation	تدل هذه الخاصية على ما إذا كان من الممكن استخدام طريقة Update أو ما إذا كان يجب عليك استخدام طرق Delete وInsert بالتحديد.

أحداث rdoConnection

يوجد ثلاثة أحداث يتم إطلاقها عند عمل وصلة (راجع الجدول ١٥-٥) وقد لا تقوم باستخدام هذه الأحداث في الغالب، ولكن قد تكون بحاجة إليهم إذا كنت تتحكم بعناية في بيئة الدخول الخاصة بك. على سبيل المثال، يتيح لك حدث BeforeConnect اختبار وحتى تغيير سلسلة الاتصال قبل عمل الوصلة. يتيح لك حدث Connect اختبار الوصلة وإعادة المحاولة في حالة حدوث أية مشاكل. ويعد هذا الحدث مفيد أيضاً في حالة استخدام أي وصلة عير متزامنة لأنه يقوم بالتعليم عند تمام العملية سواء بالنجاح أو الفشل. والشيء الوحيد الذي لا يمكنك فعله هو وقف عملية الوصل أو تقصير مدة انتهاء التطبيق. يوضح النصف الثاني من الجدول ١٥-٥ أن الأحداث التي سوف يتم إطلاقها عند تنفيذ الاستعلام.

ملاحظة يتم إطلاق حدث QueryComplete لأي إكمال استعلام وليس فقط الاستعلامات غير المتزامنة

الجدول ١٥-٥: أحداث وحدة rdoConnection

اسم الحدث	البيان
BeforeConnect	يتم إطلاق هذا الحدث قبل أن تقوم RDO بالمحاولة في الاتصال بقاعدة البيانات لأوراكل.
Connect	يتم إطلاق هذا الحدث بعد أن تقوم RDO بإكمال عملية الوصلة، حتى إذا كانت غير ناجحة.
Disconnect	يتم إطلاق هذا الحدث بعد إغلاق الوصلة عن قاعدة البيانات.
QueryComplete	يتم إطلاق هذا الحدث عند تشغيل الاستعلام على هذه الوصلة التي تم إكمالها.
QueryTimeout	يتم إطلاق هذا الحدث بعد الوصول إلى مدة QueryTimeout
WillExecute	يتم إطلاق هذا الحدث قبل تنفيذ الاستعلام، ليتيح مراجعة أو تغيير الاستعلام أو حتى إلغاءه.

ليس من الممكن استخدام الأحداث في الجدول ١٥-٥ مثلما نفعل مع معظم الوحدات الأخرى التي نستخدمها، مثل عداد الوقت أو مربع النص. وتعتبر مكتبة RDO وحدة Active X وعلينا أن نأخذ بعض الخطوات الخاصة لجعل التطبيق الخاص بنا مدرك للأحداث. والشيء الأول الذي علينا فعله هو تعريف الوحدات الموجودة لدينا، مثل وحدة rdoConnection. مع الكلمة الأساسية WithEvents. ويقوم هذا بالتعريف أنه يمكن للوحدة الاستجابة للأحداث من RDO (راجع التعليمات البرمجية ١٥-٥). لاحظ أنه لا يمكنك استخدام الكلمة الأساسية New في نفس التعريف مثل WithEvents.

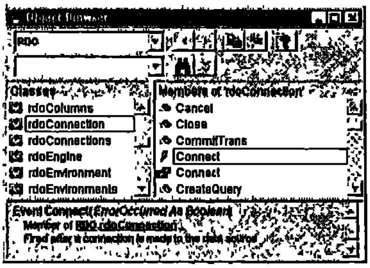
ومع كمال التعريف، علينا الآن تكوين الإجراء الذي سوف يتم استدعاءه بواسطة الحدث. ولإجراء هذه الخطوة، نقوم بتكوين إجراء باستخدام اسم rdoConnections وإضافته إلى اسم الحدث من الجدول ١٥-٥ منفصلة بخط وسيط. على سبيل المثال، إذا كان اسم الوصلات cn، فعليك إذا تلقي حدث QueryComplete وتكوين إجراء cn_QueryComplete.

تعليمات برمجة ١٥-٥

```
Private WithEvents cn As rdoConnection
Private Sub cn_Connect(ByVal ErrorOccurred As Boolean)
   If ErrorOccurred Then
      MsgBox "You encountered an error trying to connect !"
   Else
      MsgBox "You have connected !"
   End If
End Sub
```

عند استخدام الكلمة الأساسية WithEvents، عليك استخدام المربعات المنسدلة للأحداث في نافذة Editor لتوضيح النموذج الدقيق لإجراء الحدث. إذا كان الإجراء الخاص بك وتعريف Active X الخاص به لا يتطابقان معاً بدقة، سوف تحصل على خطأ مترجم. وهذه فرصة جيدة لاستخدام Object Browser المتاح أسفل قائمة View لعرض الأحداث الممكنة والمعاملات المطلوبة لهم. يوضح الشكل ١٥-٣ Object Browser الذي يتم استخدامه لعرض حدث Connect.

الشكل ١٥-٣



يقوم مستعرض الوحدة بعرض حدث Connect

العمل مع rdoResultsets

يعتبر استرداد صفوف البيانات من قاعدة البيانات لأوراكل من أهم العمليات الأساسية التي سوف تقوم بتنفيذها. فكل طريقة للوصول إلى البيانات لها الوسيلة الخاصة بها في إجراء الاسترداد، وفي RDO، يتم تنفيذ العملية بواسطة طريقة OpenResultset لوحدة Connection. وصيغة هذا الأمر كالتالي:

```
Set rs = cn.OpenResultset (source, type, locktype, options)
```

ويتم توضيح معاملات هذه الطريقة أدناه:

Source: يحدد معامل المصدر كيفية تكوين مجموعة النتائج.وسوف يكون هذا سلسلة SQL بشكل عادي التي تتسم بالترميز الدائم وتحسين وضعها بواسطة البرنامج تعمل طريقة OpenResultset أيضاً مقابل rdoQuery، الذي سوف نشاهده فيما بعد بهذا الفصل. لاحظ أنه لا يمكن أن يكون الإجراء المخزن لأوراكل لأن أوراكل لا يستطيع استرجاع مجموعات النتائج من الإجراءات المخزنة، مع أنك قد تريد الاستفادة من قسم "استرجاع جداول SQL" فيما بعد بهذا الفصل.

Type: يحدد هذا المعامل نوع الخلية التي تريد إنشائها. يدرج الجدول ١٥-٦ زيادة الأنواع المتاحة من الخلايا.

Locktype: يحدد هذا المعامل نموذج التزامن الذي يتحكم في مظاهر إقفال rdoResultset. وأنواع الإقفال المتعددة مدرجة في الجدول ١٥-٧.

Options: الخيارات الوحيدة التي يمكنك اختيارها هي rdAsynchEnable الذي يتيح لك تكوين مجموعة النتائج الخاصة بك بدون الانتظار إلى أن تكتمل (التي سوف نشاهدها فيما بعد)، وrdExecDirect الذي يكون لمعامل خادم SQL فقط للتحكم في الإجراءات المخزنة مؤقتاً.

ملاحظة يتم إجراء التحكم في قيمة QueryTimeout من خلال وحدة rdoConnection وليس وحدة rdoResultset.

الجدول ١٥-٦: القيمة المتاحة لمعامل Type.

الثابت	القيمة	البيان
RdOpenForwardOnly	صفر	يقوم هذا الثابت بفتح وحدة القراءة فقط rdoResultset. وهذا الثابت مرتبط بشكل عام مع rdoResultset الخاصة بالقراءة فقط وتعتبر أكفأ خلية (وهذه هي القيمة الافتراضية)
RdOpenKeyset	١	يقوم هذا الثابت بفتح keyset rdoResultset. وهذا النوع هو الذي سوف تقوم باستخدامه بشكل عام لمجموعة النتائج القابلة للتحديث.

الجدول ١٥-٦: القيمة المتاحة لمعامل Type.

الثابت	القيمة	البيان
RdOpenDynamic	٢	يقوم هذا الثابت بفتح وحدة rdoResultset الحيوية غير المتاحة لمستخدمي أوراكل.
RdOpenStatic	٣	يقوم هذا الثابت بفتح وحدة rdoResultset ثابتة. وهذا هو النوع الذي سوف نستخدمه بشكل عام مع البيانات ذات خاصية القراءة فقط والقابلة للتمرير.

لجدول ١٥-٧: القيم الممكنة لمعامل Locktype.

الثابت	القيمة	البيان
RdConcurReadOnly	١	يحدد هذا الثابت rdoResultset للقراءة فقط (وهذه هي القيمة الافتراضية).
RdConcurLock	٢	يقوم هذا الثابت بتحديد التزامن المقفل (الإقفال على طريقة Edit). وهذا الثابت غير مؤثر في أوراكل وقد تتلقى تحذير بتضارب الاستحداث.
RdConcurRowVer	٣	يقوم هذا الثابت بتحديد التزامن المفتوح على أساس عمود Time_Stamp (غير متاح في أوراكل).
RdConcurValues	٤	يقوم هذا الثابت بتحديد التزامن المفتوح على حسب قيم الصف.
RdConcurBatch	٥	يقوم هذا الثابت بتحديد التزامن المفتوح مستخدماً تحديثات طور Batch (يحتاج مشغل المؤشر ClientBatch). يتم استرجاع قيم الحاجة لكل صف تم استحداثه بنجاح.

من الممكن أن تكون المعلومات المطلوبة لوصف كيفية استرجاع مجموعة من البيانات مضللة، وخاصة إذا كنت حديث على البيئة. وعلى وجه الدقة أي مجموعة من الخيارات يفضل استخدامها مع أوراكل؟ لحسن الحظ، لا يمكن أن يدعم أوراكل العديد من الخيارات المتاحة،

وبالتالي من الممكن تخفيض كل هذه الأشياء إلى اثنين من الإمكانات البسيطة مجموعات النتائج القابلة للتحديث ومجموعات النتائج للقراءة فقط.

ملاحظة لاحظ أن مع RDO، لا توجد لنا حاجة إلى تحديد SQLPassThrough فلا توجد معالجة محلية مع RDO وبالتالي لا يوجد داعي للخيار. يتم معالجة كل جمل SQL بواسطة قاعدة البيانات الطرفية.

تكوين مجموعات نتائج قابلة للتحديث

أوضح طريقة لتحديث البيانات هي طريقة وحدة rdoResultset القابلة للتحديث. وعندما تقوم بإعادة الاستدعاء من المناقشة حول مؤشرات ODBC في الفصل ١١، "تقنية جانب المزود"، فنحن مقيدين قليلا باستخدام مؤشر Keyset. يوضح المثال في التعليمات البرمجية ١٥-٦ اشهر طريقة لإجراء هذا. وعليك التأكد أنك تقوم باستخدام مكتبة الخلية ClientBatch عند إعداد وحدة rdoEngine لأنها أكثر كفاءة في دعم أي خلايا قابلة للتمرير من مكتبة خلية ODBC الافتراضية.

تعليمات برمجة ١٥-٦

```
Dim rs as rdoResultset
Dim strSQL as string
strSQL = " SELECT * FROM video_store.movies WHERE rating = 'G'"
Set rs = cn.OpenResultset (strSQL, _
                    rdOpenKeyset, _
                    rdConcurValues)
```

يقوم هذا المثال بفتح rdoResultset باستخدام استعلام المصدر الموضوع في سلسلة strSQL. ونقوم بفتح خلية keyset لتتيح لنا تحديث البيانات. ونستخدم معامل locktype rdConcurValues للتحديث المفتوح. ويمكنك استخدام نوع التزامن rdConcurBatch الذي ينتظر بشكل أساسا حتى تقوم بتنفيذ طريقة BatchUpdate قبل أن تقوم بتوصيل أي تغييرات إلى أوراكل كما يمكنك أيضا استخدام نموذج rdConcurReadOnly باستخدام خلية Keyset ولكن سوف يكون كل هذا استهلاك للموارد.

ملاحظة تذكر أن rdoResultset لن تكون قابلة للتحديث إذا لم يوجد مفتاح فريد على الجدول أو إذا كنت تقوم باستخدام خلية للقراءة فقط. وإذا لم يوجد لديك امتيازات كافية لتحديث البيانات، فسوف تحصل على رسالة خطأ ثانية من طريقة Update.

تكوين مجموعات نتائج للقراءة فقط

لا تعتبر الخلايا القابلة للتحديث التي شاهدناها في القسم السابق الطريقة الكافية لتحديث البيانات في قاعدة بيانات أوراكل الخاصة بك. فالخلية ذاتها تأخذ الموارد على ماكينة المزود "الذاكرة والتشغيل" ولكنها أيضاً تأخذ الموارد على الخادم ولكنها لا تعد أيضاً منهجية قابلة للتدرج، والطريقة الأفضل هي استخدام أرخص خلية ممكنة. ويتم تحديد تكلفة المؤشر بلغة الموارد التي تقوم باستخدامها. ومع RDO، من الممكن استخدام خلية للقراءة فقط أو خلية الإرسال فقط. وإذا شاهدت الجدول ١٥-٦ والجدول ١٥-٧، سوف تلاحظ أن افتراضات النوع الأرخص من دعم الخلايا، الإرسال فقط.

مجموعات النتائج القابلة للتمرير

لتكوين وحدة rdoResultset للقراءة فقط قابلة للتمرير بالكامل (بمعنى للأمام وللخلف)، عليك استخدام التوليفة الصحيحة من الخيارات: وهذه هي الخلية الثابتة ونموذج التزامن للقراءة فقط. ومثال حول كيفية إجراء هذا موضحة في التعليمات البرمجية ١٥-٧. يمكنك استخدام مكتبة خلية Clientbatch في هذه الحالة لأنها تعطيك في هذه الحالة خلايا تمرير ذو كفاءة أكثر من مكتبة خلايا ODBC.

تعليمات برمجية ١٥-٧

```
Dim rs as rdoResultset
Dim strSQL as string
strSQL = " SELECT * video_store.movies WHERE rating = 'R'"
Set rs = cn.OpenResultset (strSQL, _
                    RdOpenStatic, _
                    RdConcurReadOnly)
```

مجموعات نتائج القراءة فقط

أرخص نموذج خلية هو الإرسال فقط، خلية القراءة فقط. وهذه الخلية هي الأرخص لأن هذه الطريقة على وجه التدقيق التي يقوم فيها أوراكل باسترجاع صفوف جملة SELECT لك بدون تمرير للخلف أو إمكانية تحديث. ولا يوجد داعي لحفظ أي شئ على المزود، على الرغم أن التطبيق الخاص بك قد يضاعف هذه الفعالية بواسطة وضع البيانات في شبكة أو في تحكم ما مشابه. يوضح المثال في التعليمات البرمجية ١٥-٨ كيفية إجراء هذا. وفي هذه الحالة لست بحاجة إلى مكتبة تمرير للخلية، لأن هذا سوف يكون استهلاك للموارد لطلب واحد (بافتراض أنك لا تستخدم خلايا تمرير في أي مكان آخر في التطبيق الخاص بك). عندما تقوم بتكوين وحدة rdoEngine، عليك تحديد خيار rdUserNone.

تعليمات برمجة ١٥-٨

```
Dim rs as rdoResultset
Dim strSQL as string
StrSQL = " SELECT * video_store.movies WHERE rating = 'R'"
Set rs = cn.OpenResultset (strSQL, _
RdOpenForwardOnly, _
                RdConcurReadOnly)
```

قد يبدو من التعقيد التدقيق في هذه الأمثلة التي تختلف في سطر واحد فقط، إلا أنها توضح كل أنواع الخلايا الممكنة التي يمكنك العمل معها في RDO. وبتكوين وحدة rdoResultset لست بحاجة بعد الآن للتورط في تعقيد أكثر.

ملاحظة يمكنك كمستخدم لأوراكل، أن تغفل كل شئ سمعته عن استرداد مجموعات نتائج متعددة بواسطة RDO. وتعتبر مجموعات النتائج المتعددة ميزة في خادم SQL الذي تقوم فيه جملة SELECT في الإجراء المخزن بإرسال مجموعة النتائج المتعلقة ثانية إلى برنامج الاستدعاء. ولا يقوم أوراكل بالعمل بهذه الطريقة: فلا يمكنك حتى استرجاع مجموعة نتائج منفردة من أوراكل ما لم تستخدم وحدات أوراكل الخاصة بـ OLE (راجع الفصل الثامن عشر). يمكنك الآن مضاعفة هذا التأثير بواسطة استرداد جدول PL/SQL إلى واحد أو أكثر من rdoResultsets. وسوف نقوم بتغطية هذا فيما بعد بهذا الفصل. وهذه تعد كمية عمل كبيرة، ولكنها على الأقل ممكن إنجازها.

مكتبة خلية ClientBatch

لقد ذكرنا مكتبة خلية ClientBatch أكثر من مرة، ولكن تستحق اهتمام خاص. لقد تم تكون هذه المكتبة بواسطة فريق عمل FoxPro للفيجوال بيسك على وجه الخصوص وتقدم مكتبة خلية أسرع بكثير من مكتبة ODBC الافتراضية. ومع أن مكتبة خلية ClientBatch تأتي على حساب أثر أكبر قليلاً، إلا أنه عليك استخدام هذه المكتبة دوماً إذا كنت تستخدم مجموعات النتائج القابلة للتمرير، سواء كانوا ذو قابلية للتحديث أو لا.

ومع هذا فإن إمكانات المكتبة تعد متطورة جداً. فكما يدل الاسم، يمكنك استخدام المكتبة لعمل تحديثات بالدفعة للتغييرات. وهذا يعني أنه يمكنك توقع تحديث أكثر من معدل صف بعد الآخر، فمن الممكن حفظ كل التحديثات في قاعدة البيانات لأوراكل وجمعهم معاً. لا تخلط بين التحديثات بالدفعة مع التحكم في الحركة، التي لا تزال في حاجة لاستخدامها بالإضافة إلى التحديثات بالدفعة.

من الممكن أن يعني استخدام التحديثات بالدفعة حفظ التغييرات الخاصة بك لدقائق أو حتى لساعات الأمر الذي من الممكن أن يسبب بعض المشاكل عندما يحين الوقت لإجراء التحديث. ولا تعد هذه الطريقة بالتأكيد المناسبة لتغيير البيئة بشكل هائل وكذلك فهي غير قابلة للتدرج، ومع هذا فقد تجد هناك حالات تكون مناسبة لها. وإذا قمت باستخدامها، فعليك إذاً التأكد أنك تقوم بإعدادها بشكل صحيح، وعليك وضع النقاط التالية في إعتبارك:

- تحديد مكتبة خلية ClientBatch للوصلة قبل إعداد rdoResultset.
- تحديد نموذج التزامن rdConcurBatch.
- استخدام طريقة BatchUpdate لحفظ أي تغييرات دفعة.
- حل أي تعارضات أو أخطاء تحديث.

يوجد عمل كثير لابد من تنفيذه للحصول على تحديثات دفعة للعمل بها وخصوصا عندما تأتي لحل تضارب أو خطأ. ومن الحكمة أخذ ورقة من دفتر تصميم الاستنساخ وأن تحاول التصميم بحيث لا توجد إمكانية تضارب التحديثات. والتصميم بهذه الطريقة قد يكون شاق ولكنه ممكن. على سبيل المثال، قاعدة بيانات الموارد البشرية بها مستخدم واحد مسئول عن تحديث المستخدمين الذين لهم حق الوصول إلى عروض محددة أو قد يكون مستخدم واحد له التحكم في مجموعة فرعية محددة من البيانات في معدل وقت محدد. ويمكنك التحديد حسب احتياج التطبيق الخاص بك لهذه المتطلبات.

استخدام أحداث rdoResultset

مثلما يحدث مع وحدة rdoconnection تعترض وحدة rdoResultset الأحداث التي يمكنك استخدامها مع الكلمة الأساسية WithEvents في تعريف rdoResultset. وهذه الأحداث مضمنة في الجدول ١٥-٨. تضع معظم هذه الأحداث وسائل تحكم أكثر في عمليات rdoResultset في متناولك.

الجدول ١٥-٨: إطلاق الأحداث عند استخدام rdoResultset.

اسم الحدث	البيان
Associate	يظهر هذا الحدث بعد ربط الوصلة الجديدة بالوحدة.
Dissociate	يظهر هذا الحدث بعد إعداد الاتصال على لا شيء.
Resultschange	يظهر هذا الحدث بعد تغيير مجموعة الصفوف الحالية وينطبق هذا على مجموعات النتائج المتعددة المستخدمة لاسترداد جداول PL/SQL.

الجدول ١٥-٨: إطلاق الأحداث عند استخدام rdoResultset.

اسم الحدث	البيان
RowStatusChange	يظهر هذا الحدث بعد تغيير مجموعة الصفوف الحالية بواسطة طريقة Edit أو Delete أو AddNew.
RowCurrencyChange	يظهر هذا الحدث بعد تغير مؤشر الصف الحالي.
WillAssociate	يظهر هذا الحدث قبل ربط وصلة جديدة بالوحدة (يتيح لك إلغاء العملية).
WillDissociate	يظهر هذا الحدث قبل إعداد الاتصال على لا شيء (يتيح لك إلغاء العملية).
WillUpdateRows	يظهر هذا الحدث قبل حدوث تحديث في قاعدة البيانات.

يوجد نوعان هامان من الأحداث هما: أولاً أربعة أحداث مرتبطة بتشغيل rdoRsultsets المنفصلة وتجد rdoRsultsets المنفصلة هذه في مكان إعداد وحدة rdoRsultset المستقلة بذاتها (مع كلمة الأساس الجديدة) فعليك ملئها ثم إغلاق الوصلة بجملة Set cn = Nothing. وسوف تظل rdoRsultset نشطة وفي حالة استعداد لإعادة الاتصال بواسطة خاصية ActiveConnection. ويعد هذا مفيد أساساً إذا قمت بالتشغيل في طور ClientBatch وكذلك لحفظ أي تغييرات حتى تقوم بإعادة الاتصال وعمل التغييرات في قاعدة البيانات بطريقة BatchUpdate (التي تطابق مع الأساليب المستخدمة مع واجهة الويب).

الحدث الثاني الهام هو WillUpdateRows. ويتم تنفيذ هذا الحدث قبل أن تقوم Rdo بمحاولة إرسال التغييرات إلى قاعدة بيانات أوراكل باستخدام طريقة Update. يتيح لك الحدث الفرصة لاستخدام الطرق الخاصة بك في تحديث البيانات ومع وجه الخصوص باستخدام الإجراءات المخزنة. وتصبح هذه ميزة مفيدة في أتاحتها لك معالجة التحديثات بواسطة مجموعة نتائج القراءة فقط، لأنه يمكنك تزويد تعليمات البرمجة رجوع للدلالة على الإكمال الناجح، ولن يكون من الضروري تجديد مجموعة النتائج.

ولا يوجد أي من هذه الأحداث متاح افتراضيا وعليك تعريف rdo Resultset باستخدام كلمة الأساس WithEvents وترميزها على نوح ما شاهدنا في التعليمات البرمجية ١٥-٥ عندما فعلنا نفس الشيء مع وحدة rdo Connection.

اكتشاف خصائص وأساليب rdo Resultset

تم إنشاء RDO بعد DAO وتستخدم نفس أسماء الطرق بالرغم من اســتخدام "rdo Resultset" بدل من "Recordset" الخاصة بــ DAO وسوف يقول مصمم الــ Object-Oriented أن هذه سوف تمنح تعدد الأشكال التي يمكن أن يتطابق فيها نفس اسم الطريقة مع وحدات مختلفة ضمنياً. وسوف نرى أن نفس الشيء سوف ينطبق على وحدات بيانــات Active X ووحــدات أوراكــل الخاصة بــ OLE. ومع أنه من الأفضل معرفة أنه يوجد سبب نظري لتســمية الطريقــة بنفــس الاسم، إلا إنه يوجد رأي عملي أيضاً يقول لماذا نسمي الطرق بأي شيء مختلف؟، ويستثنى مــن هذا مصطلح rdo Resultset.

ملء rdo Resultset

كما هو الحال مع DAO، لا يتم ملء rdo Resultset كلية بالبيانات حتى يتم الوصــــول إلـــى الصف النهائي. ومع أي خلية قابلة للتمرير، لا بد أن يكون هدفك هـــو الوصـــول إلـــى نهايــة مجموعة النتائج أو من الأفضل، إغلاق مجموعة النتائج بأسرع ما يمكن. ولأنــــه مــن الممكــن لأوراكل دعم مجموعات النتائج المعلقة (بمعنى مجموعات النتائج التي لم يتم إغلاقها)، ولا تعــد هذه مشكلة خطيرة على وجه الخصوص وكل ما عليك محاولته هو إغلاقهم بقــدر مــا أمكــن. وتعتبر RDO أكثر كفاءة في ملء مجموعة النتائج مقارنة مع DAO مــع عــدم وجــود طــرق Fillcache.

تغيير الصف الحالي

يمكنك استخدام أساليب DAO Move التي شاهدناها أولاً في الجدول ١٤-٦ لنقل الصف الحــالي في rdo Resultset. ولا بد من اختبار خصائص BOF و EOF الخاصتان بــ rdo Resultset كم فعلت مع DAO. والاختلاف الرئيسي عن DAO هو أن RDO تتيح لــك اســتخدام أسـلوب Movelast الغير متزامن. وفي هذه الحالة، يمكنك الــترتيب لحــدث Completion مــن rdo Resultset عندما يكون النقل تام. وهذا مفيد لــ rdo Resultset الكبيرة التي تستغــرق وقــت طويل في عملية الملء، وعلى كلٍ، فهذه إشارة تحذير بأنه قد يوجد لديك مجموعات نتائج كبــيرة جدا. ولاحظ أنه إذا كان يوجد لديك خلايا للإرسال فقط فإنه يمكنــك اســتخدام أسـلوب Move Next.

الوصول إلى حقول rdo Resultset

تحتوي Recordset الخاصة بوحدات وصول البيانات على مجموعة Fields بينما تحتــوي rdo Resultset على مجموعة rdo Columns. والاستخدام الرئيسي لهذه المجموعة هــو مراجعــة

البيانات في Resultset، ولكن سوف نشاهد فيما بعد أن هذه المجموعة يمكن استخدامها لعرض هيكل جدول قاعدة البيانات. وقد تحتاج أحياناً إلى نوع المعلومات الذي يدل على أن هذه المجموعة من الممكن أن تعطيك الكثير حول استخدام rdo Queries والوحدات الأخرى. على سبيل المثال، يمكنك اختيار خاصية Chunk Reqeuired لمشاهدة ما إذا كنت تريد استخدام أساليب Get Chunk و Append Chunk في أعمدة البيانات الطويلة والثنائية.

وكما هو الحال مع DAO، للبحث عن قيمة أي عمود في الصف الحالي من rdo Resultset. يمكنك الإشارة إليها عن طريق الفهرس (العدد حسب الترتيب) أو حسب الاسم. والصيغة الخاصة بذلك هي:

```
rs (iCol)
```

أو

```
rs ("name")
```

كما أننا سوف نستخدم هذه المعلومات لملء مربع القائمة كما هو الحال مع DAO. وفي هذا الإصدار سوف نستخدم أسلوب أكثر كفاءة باستخدام With ... End With للإشارة إلى rdo Resultset كما هو موضح في التعليمات البرمجية ١٥-٩. نعم توجد طرق محددة للبيانات لإجراء هذا الأسلوب، ولكن التعليمات البرمجية توضح المفهوم أكثر.

تعليمات برمجة ١٥-٩

```
Public Sub FillListBox(ListBox As ComboBox)
' Fill a list box with movie names
' Ignore error trapping
   Dim rs As rdoResultset
   Dim strSQL As String

   strSQL = "SELECT movie_name FROM video_store.movies"
   Set rs = cn.OpenResultset(strSQL, _
                     rdOpenForwardOnly, _
                     rdConcurReadOnly)
   ListBox.Clear
   With rs
      While Not .EOF
         ListBox.AddItem rs("movie_name")
         .MoveNext
      Wend
   ListBox.ListIndex = 0 ' Display the first item
   End With
   Set rs = Nothing
End Sub
```

تغيير الصفوف والقيم في rdo Resultset

وباتباع مسار DAO ثانية، نستخدم أساليب Add New وEdit وUpdate وDelete في rdo Resultset القابلة للتحديث إذا كنا نريد تغيير قيم الصف في rdo Resultset وسوف نشاهد مثال لذلك فيما بعد بهذا الفصل بقسم "التجميع معا".

البحث عن صف بعينه

سوف تواجه مشكلة واحدة مع RDO وهو أنها لا تدعم أي من أساليب Find. وسوف يكون هذا مزعج لهؤلاء الذين تطوروا على Resultset القابلة للتحديث في JET وRDO. ولا يوجد أي شيء لهذه الإمكانية على وجه الخصوص يمكنها القيام به باعتبارها متصلة بقاعدة البيانات البعيدة. وكما ذكرنا في الفصل الأخير، إذا قمت باستخدام أي من أوامر Find، فهذا يدل على أنه

لابد أن تكون قد قمت بتحديد مجموعة نتائج أصغر في البداية. وإذا وجدت صعوبة في إيقاف الأمر، عليك تقليل المعايير التي تقوم باستخدامها لجملة Find وعليك بناء rdo Resultset جديدة حولها. وبشكل غريب، تقدم وحدات أوراكل الخاصة بـ OLE إمكانات FND بشكل أكثر من RDO.

سحب صفوف متعددة

تحتوي RDO على أسلوب Getrows الذي شاهدناه في الفصل السابق. يتيح لك هذا الأسلوب سحب صفوف متعددة من قاعدة البيانات لأوراكل إلى مصفوفة المتغيرات. ويعتبر هذا الأسلوب واحد من أكفأ الأساليب المستخدمة في استرداد البيانات إذا كنت بحاجة إلى كل الصفوف والأعمدة.

الإغلاق rdo Resultset

يجب إغلاق rdo Resultset على وجه السرعة وعليك فعل هذه الخطوة بواسطة إعداد الوحدة إلى لا شيء مثل.

```
Set rs = Nothing
```

أو يمكنك استخدام أسلوب Close.

استخدم وحدات rdoQuery

يعمل تكوين rdo Resultset مع سلاسل SQL الثابتة كما شاهدنا في الأقسام السابقة بشكل جيد ولكن إلى حد معين. فهل نظرت من قبل إلى الاستعلامات في تعليمات البرمجة الخاصة بك ولاحظت أنك غالباً تقوم بتنفيذ نفس الاستعلام مع تغيير طفيف في عبارة WHERE بين كل واحد وآخر؟ على سبيل المثال، قد تقوم بترميز

```
SELECT * video_store.movies WHERE rating = 'PG-13'
```

وفيما بعد تقوم بتشغيل نفس الاستعلام بواسطة

```
SELECT * video_store.movies WHERE rating = 'G'
```

وبالتأكيد يوجد لديك تعليمات برمجة تقوم بإنشاء استعلامات أكثر تعقيداً من هذه الاستعلامات، قطعة بقطعة. وقد يتطلب هذا معالجة أساسية لتجميع الاستعلام معاً. ولكن توجد مشاكل عديدة في هذه العملية. أولاً، من الممكن أن تصبح عملية إنشاء الاستعلام معقدة مما يجعل صعوبة في قراءة تعليمات البرمجة. ثانياً لا بد أن تقوم DO و ODBC وأوراكل بتحليل الاستعلام في كل مرة تقوم بتنفيذه.

من الممكن أن تزود وحدة rdo Query بحل جيد لهذه المشكلة بالرغم من وكما سترى في القسم التالي، أنها أكثر أهمية في تنفيذ الإجراءات المخزنة. وتعتبر rdo Queries طريقة متقدمة جداً في التوصيل بـ ODBC وقاعدة البيانات بما أنهم يستفيدوا من إمكانات معامل ODBC.

ملاحظة لا تستخدم RDO معاملات الربط لأوراكل. فقط وحدات أوراكل الخاصة بـ OLE يمكنها استخدامهم وسوف نشاهد هذا في الفصل الثامن عشر.

أولاً سوف ننظر إلى كيفية استخدام rdoQuery في طريقة بسيطة باستخدام ما يسمى سلسلة SQL المتسمة بالترميز الدائم الذي نستخدمه حتى الآن مع أسلوب Resultset. ونحن بحاجة لاستخدام أسلوب CreateQuery الخاص بوحدة rdo Connectin الذي يحتوي على الصيغة التالية:

```
query = cn.CreateQuery (name, query_string)
```

Name: يعتبر Name اسم عشوائي للاستعلام، وتقوم query_string بتحديد سلسلة SQL. ونقوم الآن باستخدام أسلوب OpenResultset لوحدة Query لاسترجاع مجموعة النتائج من أوراكل. وصيغة هذا الأسلوب هي نفس صيغة وحدة rdo Connection ما عدا أنه في هذه الحالة لا نقوم بتزويد سلسلة SQL. راجع قسم "العمل مع rdo Resultsets" مبكراً في هذا الفصل. ولتكوين استعلام واستخدامه لإنتاج مجموعة النتائج، من الممكن استخدام أسلوب Open Resultset كما هو موضح في التعليمات البرمجية ١٥-١٠.

تعليمات برمجية ١٥-١٠

```
Dim qy As rdoQuery
Dim rs As rdoResultset
Dim strSQL As String

strSQL = "SELECT * FROM video_store.movies WHERE rating = 'PG-13'"
Set qy = cn.CreateQuery("query1", strSQL)
Set rs = qy.OpenResultset(rdOpenForwardOnly, _
                RdConcurReadOnly)
```

ومن الأفضل ترك اسم الاستعلام فارغ وخاصة عند تطوير تعليمات البرمجة. وإذا قمت بتحديد الاسم، فلن يمكنك آنذاك إعادة تكوين الاستعلام. وفي بيئة الإنتاج، لا يفضل حدوث هذا على أي وجه ولكن ترك اسم الاستعلام فارغ يعني أنه يمكنك إعادة تنفيذ التكوين للاستعلام (بواسطة نقل مؤشر سطر البرنامج الحالي للخلف) ثم تقوم بتحديد أي مشاكل تواجهك.

تلميح يعتبر نقل مؤشر سطر البرنامج الحالي واحد من أهم السمات المفيدة والغير معلن عنها بشكل كافي للشخص. وفي طور Development، عليك عرض تعليمات البرمجة الذي تقوم بتشخيصها، ثم خطف السهم الصغير الأصفر الكائن بالهامش الأيسر وسحبه لأعلى. يجب أن تقابل هذه الخاصية بند قائمة Step Back.

ولتطوير المثال البسيط الموضح في التعليمات البرمجية ١٥-١٠، نقوم بتقديم مفهوم معاملات الاستعلام. ولكشف ما يجب عليك استخدامه كمعامل، عليك النظر إلى تعليمات البرمجة الخاص بك للبحث عن القيم التي تتغير. وفي المثال أعلاه، إنه عمود Rating، وهذا في الاستعلام المعقد، الذي يمكنك الحصول من خلاله على قيم عديدة من الممكن أن تتغير. توجد مجموعة rdoParameters لكل rdoQuery، وتزود هذه المجموعات بطريقة إعداد استعلام مستقل عن هذه القيم في عبارة WHERE، وتقوم بتزويد القيم التي تتغير فقط قبل أن تقوم بتنفيذ الاستعلام.

ولجعل ODBC يعرف المكان الذي تمضي إليه هذه المعاملات في الاستعلام، عليك وضع علامة استفهام في سلسلة SQL كمتغير تعويض للمعامل. وهذه هي الصيغة التي تستخدمها API الخاصة بـ ODBC من الداخل. وفي المثال، سوف نستخدم

```
strSQL = "SELECT * video_store. Movies WHERE rating = ? "
```

والآن علينا أن نزود قيمة لمتغير تعويض هذا، ولهذا نستخدم مجموعة المعاملات. وليس من اللازم تكوين أو إضافة وحدات معامل إلى المجموعة، لأن RDO تقوم بتكوينهم لنا عند تكوين الاستعلام. وما علينا القيام به هو تزويد قيمة لهم. توضح التعليمات البرمجية ١٥-١١ مثال حول كيفية إجراء هذا في الاستعلام باستخدام معاملين واحد سلسلة وآخر رقم. ولاحظ أنه عليك الحصول على المعاملات في الترتيب الصحيح لهم وأنهم مرقمين على التتابع من الصفر.

تعليمات برمجة ١٥-١١

```
Dim qy2 As rdoQuery
Dim rs As rdoResultset
Dim strSQL As String
```

```
strSQL = "SELECT * video_store. movies WHERE year_of_release > ? "
strSQL = strSQL & " AND rating = ? ORDER BY movie_name"
Set qy2 = cn.CreateQuery("", strSQL)
qy2.rdoParameters(0) = 1995
qy2.rdoParameters(1) = "PG-13"
Set rs = qy2.OpenResultset(rdOpenForwardOnly, _
rdConcurReadOnly)
While Not rs.EOF
  MsgBox rs(1)
  rs.MoveNext
Wend
Set rs = Nothing
```

وإذا كنت تريد تغيير أي من هذه المعاملات، فلست بحاجة إلى إعادة بناء الاستعلام ولكن عليك فقط تغيير المعاملات التي تريدها ثم استخدام أسلوب Requery الخاص بـ rdoResultset ويتم آنذاك تنفيذ العمل. وعلى كل، تأكد أنك لا تقوم بإعادة تكوين متغير الاستعلام مرة ثانية حتى تكون على دراية بالغرض من هذا العمل.

تعرض وحدة rdoQuery منوعات مختلفة من الخصائص، وهذه الخصائص موضحة في الجدول ١٥-٩. تعد هذه الخصائص مفيدة على وجه الخصوص في تكوين rdoQueries مستقلة بذاتها التي تقوم فيها بإعداد الوحدة ثم تزويد الخصائص بعد ذلك، والأخير منها يقوم بإعداد ActiveConnection إلى rdoConnection صحيح.

الجدول ١٥-٩: خصائص RdoQuery.

الخاصية	البيان
ActiveConnection	تقوم هذه الخاصية بإرجاع أو وضع الوصلة النشطة لوحدة الاستعلام هذه. وتحتاج لهذا إذا كنت قد قمت بتكوين وحدة rdoQuery المستقلة بذاتها.
BindThreshold	تشير هذه الخاصية إلى أن العمود الأكبر لابد أن يكون منضم آليـاً والقيمة الافتراضية هي ١٠٢٤، ولأي شئ أكبر من قيمة البدء، عليك استخدام أساليب GetChunk وAppendChunk.

الجدول ١٥-٩: خصائص RdoQuery.

الخاصية	البيان
KeysetSize	تشير هذه الخاصية إلى حجم الذاكرة الوسيطة keyset المستخدمة عند تكوين الخلايا.
LockType	تقوم هذه الخاصية بإرجاع أو تحديد نوع التزامن، الذي يقوم بتحديد أسلوب Locking للاستعلام.
MaxRows	تشير هذه الخاصية إلى عدد الحد الأقصى من الصفوف التي يجب إرجاعها بواسطة الاستعلام.
Prepared	تشير هذه الخاصية إلى ما إذا كان يجب تجهيز وحدة rdoQuery بواسطة وظيفة SQLPrepare الخاصة بـ ODBC.
QueryTimeout	تشير هذه الخاصية إلى المدة التي ينتظرها Driver Manager الخاص بـ ODBC قبل انتهاء وقت الاستعلام وظهور حدث QueryTimeout.
RowsAffected	تشير هذه إلى عدد الصفوف المصابة بواسطة استعلام خاص بإجراء معين.
RowsetSize	تحدد هذه الخاصية عدد الصفوف التي يتم حفظها بداخل الذاكرة الوسيطة عند إنشاء الخلية. ولابد أن يكون حجم RowsetSize "١" لخلية الإرسال فقط.
SQL	تحتوي هذه الخاصية على سلسلة SQL التي تكون مصدر الاستعلام.
StillExecuting	تشير هذه الخاصية إلى ما إذا كان الاستعلام يقوم بالتنفيذ من عدمه.
Type	تشير هذه الخاصية إلى نوع الاستعلام (مثل استعلام تحديد أو إجراء مخزن).

الجدول ١٥-٩: خصائص RdoQuery.

الخاصية	البيان
Updatable	تشير هذه الخاصية ما إذا كانت مجموعة النتائج التي قمت بتوليدها بواسطة rdoQuery يمكن تحديثها من عدمه.

ملاحظة سوف تقيد خاصية MaxRows عدد الصفوف المرتجعة من نتائج أوراكل عبر ODBC ولكن لن يكون لها أي تأثير على استعلام أوراكل على الإطلاق. وإذا كنت تريد تقييد عدد الصفوف، فعليك استخدام صيغة أوراكل في عبارة WHERE، WHERE rownum 500ذ أو شئ ما مشابه.

وأيسر طريقة لإعداد rdoQuery هو تكوين إجراء خاص مستقل في المنظومة الخاصة بك لبدء كل الاستعلامات المستخدمة في المنظومة. وعليك هنا تعريف وحدة rdoQuery كمتغير لمستوى منظومة خاص بحيث يكون متاح على كل المنظومة. تذكر أنه عليك تكوين الاستعلام مرة واحدة فقط، على الرغم أنك تقوم بتنفيذه لمرات عديدة. حاول تجنب تحديد الاستعلامات في حدث Form_Load بكل الطرق الممكنة، وعليك استدعاء إجراء بدء الاستعلام من هناك. ولن تكون في حاجة إلى جعل وحدة rdoQuery متاحة خارج المنظومة أو الطبقة أو النموذج. وإذا قمت بذلك، فمن المحتمل ألا تكون قمت بتغليف وصول البيانات بالشكل المناسب في التصميم الخاص بك، فيفضل مراجعة الحالة.

ملاحظة تعتبر وحدة rdoPreparedStatement قديمة ولقد تم استبدالها بـ rdoQueries في الفيجوال بيسك ٥. وعليك التحويل إلى rdoQueries بمجرد أن تكون لديك الفرصة.

تنفيذ أوامر SQL

يمكنك تنفيذ أوامر SQL بنفس طريقة DAO، ولكن هذه المرة نستخدم طريقة Execute من وحدة rdoConnection بدل من وحدة Database. وسوف تستخدم هذا الأسلوب بشكل عادي لتشغيل أوامر INSERT وUPDATE وDELETE ويحتمل أوامر COMMIT وROLLBACK

للتحكم في الحركة. (لاحظ أن RDO يوجد بها autocommit على كل إمكانية وصول إلى قاعدة البيانات تتضمن تنفيذ أوامر SQL، لذلك فإن COMMIT الإضافية لم يتم استدعائها افتراضياً في هذه الحالة). توضح التعليمات البرمجية ١٥-١٢ مثال حول كيفية إجراء هذا باستخدام RDO. ويمكنك تضمين أمر SQL واحد فقط ولا تقوم بتضمين فاصلة منقوطة دالة على الانتهاء. (راجع الشريط الجانبي: قطع رموز SQL المجهولة فيما بعد بهذا الفصل لطريقة أفضل لإجراء ذلك).

تعليمات برمجية ١٥-١٢

```
Dim strSQL As String
strSQL = "DELETE video_store.movies WHERE movie_id = 303"
cn.Execute strSQL
strSQL = "COMMIT"
cn.Execute strSQL
```

ملاحظة تذكر أنك إذا قمت بتنفيذ هذه الأوامر، سوف يزال الصف المحذوف في أي rdoResultset حالية حتى يتم تجديدها.

يوجد خياران يمكنك استخدامهما عن طريق أسلوب Execute الذي يختلف بشكل واضح عن DAO. وهذان الخياران موضحان في الجدول ١٥-١٠.

الجدول ١٥-١٠: خيارات تنفيذ طريقة The rdoconnection.

الثابت	القيمة	البيان
RdAsyncEnable	٣٢	يقوم هذا الثابت بتنفيذ أمر SQL الغير متزامن.
RdExecDirect	٦٤	يتجاهل هذا الثابت إنشاء الإجراء المخزن لتنفيذ الاستعلام. وهذا الثابت لا يتطابق مع أوراكل.

إذا أردت استخدام خيار غير متزامن، فسوف تكون بحاجة لاختبار إكمال الحدث كما سوف نشاهد في قسم "التشغيل غير المتزامن" فيما بعد بهذا الفصل.

ملاحظـــة - على الرغم أن أداة مثل SQL*Plus سوف تقوم بإبلاغك بعـــدد الصفـــوف المصابة من جملة DML "Action"، فلا توجد آلية لأوراكل لاسترجاع هذه القيمة إلى ODBC. ونتيجة لذلك، سوف تكون خاصيـــة RowsAffected دائما صفر.

استدعاء الإجراءات المخزنة

شاهدنا في القسم السابق كيفية تنفيذ جملة SQL في RDO. والذي نقوم بإجرائه بالفعل هو تنفيـــذ قطعة رموز PL/SQL (وفي هذه الحالة تكون جملة واحدة فقط)، وهذه هي الطريقة التي يمكنـــك من خلالها تنفيذ إجراء مخزن كما شاهدنا في الفصل الرابع عشر، وحدات الوصول إلى البيانات. وعلى كل، كما هو الحال مع DAO، فنحن مقيدين بالترميز الدائم أو جملة SQL المبنية مع عـــدم وجود طريقة لاسترجاع المعاملات الخارجة. ولكن مع أي تقنية بعد DAO، من الممكن الاستفادة من المعاملات الخارجة ويزود هذا بتطوير أساسي في طريقة استخدامنا للإجراءات المخزنة.

عليك التأكد أنه يوجد لديك مشغل ODBC المناسب للتأكد أنه يمكنك اســـترجاع المعـــاملات الخارجة ولتعريف الإجراءات الخاصة بك بمخطط قاعدة البيانات. ومشغل ODBC الذي يأتي مع الفيجوال بيسك ٦ يلبي هذه المتطلبات، على الرغم أن مشغلين كثيرين لا يقومـــوا بذلـــك (منـــهم إصدار فيجوال بيسك ٥). والشيء الوحيد المفقود في مشغل ODBC للفيجوال بيسك هو أنـــــه لا يدعم حزم PL/SQL.

وللاستفادة من المعاملات الخارجة، علينا زيارة وحدة rdoQuery مرة ثانية والتي كنـــا قـــد تعرضنا لها قبل الآن بهذا الفصل. وسوف نستخدم نفس متغيرات التعويـــض للمعـــامل (علامـــة الاستفهام) ومجموعة rdoParameters. وافتراضيا، أي معامل يعتبر معامل إدخال، الأمر الذي كان يناسب استعلام SELECT، ولكن لاستخدام المعاملات باستخدام الإجراءات المخزنـــة التـــي يوجد بها اتجاهات غير الإدخال، علينا التأكد أن الاتجاه الصحيح معد لكل معامل. ولحسن الحـــظ، يمكنك إجراء هذا باستخدام خاصية Direction لكل معامل. وفوق كل هذا يوجد أربع خصـــائص لوحدة Parameter وهذه الخصائص موضحة في الجدول ١٥-١١.

الجدول ١٥-١١: خصائص وحدة Parameter.

الخاصية	البيان
Direction	توضح هذه الخاصية اتجاه المعامل، كما هو موضح في الجـــدول ١٥-٩.

الجدول ١٥-١١: خصائص وحدة Parameter.

الخاصية	البيان
Name	توضح هذه الخاصية اسم المعامل. (من الممكن أن يترك فارغ)
Type	توضح هذه الخاصية نوع البيانات للمعامل.
Value	توضح هذه الخاصية القيمة الفعلية للمعامل.

ملاحظة لاحظ أن Value تعتبر الخاصية الافتراضية لوحدة Parameter، لذلك لا يجب عليك تحديدها. سوف تجد أيضاً أن خاصية Parameters لوحدة rdoQuery افتراضية، ولذلك لا يجب عليك تحديدها.

عليك إلقاء نظرة على الإجراء المخزن الذي تريد تنفيذه والبحث عن كل معامل (المعاملات العادية) وطور كل منه. تذكر أنه إذا لم يقم الإجراء المخزن الخاص بك بتحديد الطور، فسوف يقوم بافتراض IN. يوضح الجدول ١٥-١٢ التحويل من طور PL/SQL إلى ثابت الاتجاه لمعامل RDO.

الجدول ١٥-١٢: أطوار معاملات RDO.

طور PL/SQL	ثابت المعامل RDO	البيان
IN	RdParamInput	قيمة مدخلات فقط
IN OUT	RdParamInputOutput	قيمة مدخلات ومخرجات
OUT	RdParamOutput	قيمة مخرجات فقط

تلميح تظهر RDO 2.0 ناجحة تماماً في تحديد اتجاه ونوع بيانات المعاملات بذاتها، لذلك يمكنك دائماً إهمال خاصية الاتجاه. ومع هذا إذا قمت باستخدام نوع البيانات LONG أو -RAW، فعليك في هذه الحالة تعريف هذان النوعان بنفسك.

وكمثال، سوف نقوم بتطوير تعليمات البرمجة لتنفيذ الإجراء المخزن الذي شاهدناه في الفصل السابق (انظر التعليمات البرمجية ١٤-٧). أولا، سوف نقوم بتطوير الإجراء المخزن. وسوف نستخدم إجراء movies_insert3 (راجع التعليمات البرمجية ١٥-٣)، الذي يقوم بإدخال سجل فيلم جديد ووضع قيمة المفتاح من مسلسل أوراكل ثم يقوم بإرجاع تلك القيمة لنا عبر معامل طور IN OUT. وميزة هذا هي أنه يمكنك استخدام القيمة كمفتاح ربط بجملة INSERT أخرى. على سبيل المثال، الشخص الذي يقوم بتخزين البيانات لنسخ الفيديو من فيلم. يتيح المثال بالفعل تمرير قيمة المفتاح الخاص بك وإنشاء قيمة جديدة في حالة تمرير صفر. يعطي هذا تحكم أكثر في التطبيقات المتطورة مثل الاستنساخ الذي توجد فيه بكل طبعة قاعدة بيانات واحدة صف بمفتاح محدد وتريد تكرار ذلك في الطبعة الثانية.

تعليمات برمجة ١٥-١٣

```
PROCEDURE movies_insert3(
p_movie_id IN OUT NUMBER,
p_name      IN    VARCHAR2,
p_length    IN    NUMBER,
p_rating    IN    VARCHAR2,
p_year      IN    NUMBER) IS
/*  Insert a new movie into the movie table
If the formal parameter, p_movie_id is zero it is assigned
from the next available sequence number.
*/
BEGIN
IF p_movie_id = 0 THEN
SELECT movie_sequence_id.NEXTVAL INTO p_movie_id FROM dual;
END IF;
INSERT INTO movies VALUES (
p_movie_id, p_name, p_length, p_rating, p_year, USER, SYSDATE);
COMMIT;
EXCEPTION
/* Whatever the problem just rollback and
pass the error back to Visual Basic */
WHEN OTHERS THEN
```

```
ROLLBACK;
RAISE;
END movies_insert3;
```

لتنفيذ الإجراء المخزن هذا، يجب إعداد rdoQuery لتنفيذه. ويجب إعداد متغير تعويض معامل واحد لكل معامل في الإجراء المخزن. وليس من الممكن عمل ترميز دائم لأي من المعاملات لأن ODBC يتحسس متغير تعويض لكل معامل. ولأن افتراض كل المعاملات على اتجاه IN، يمكنك تحديد اتجاه المعاملين OUT و IN OUT بشكل صريح وعليك التأكد من صحة هذا التحديد. وعلى مدار هذا الوقت سوف نتعرض للاستعلامات بعناية أكثر ثم نقوم بتكوين إجراء CreateQueries الذي يتم استدعاءه على مستوى المنظومة لتكوين أي rdoQueries سوف تقوم المنظومة باستخدامها، (راجع التعليمات البرمجية ١٥-١٤). عندما تكون المقارنة مع تكوين نوع استعلام SELECT، يجب استخدام صيغة مختلفة، تتضمن أقواس معقودة وكلمة الأساس CALL ولكن علينا الاستمرار في الاحتفاظ متغيرات التعويض لمعامل ODBC.

تعليمات برمجية ١٥-١٤

```
Dim qyMovieInsert as rdoQuery
Private Sub CreateQueries()
'  Set up any queries required in this module
   Dim strSQL As String
   StrSQL = "{call movies_insert3 (?, ?, ?, ?, ?) }"
   Set qyMovieInsert = cn.CreateQuery("", strSQL)
End Sub
```

والآن عندما تأتي إلى تنفيذ الإجراء المخزن، تحتاج إلى إجراء الفيجوال بيسك لعمل الاستدعاء. توضح التعليمات البرمجية ١٥-١٥ طريقة واحدة لإجراء ذلك. لاحظ أن الإجراء أكثر بساطة ونظافة من عملية الترميز الدائم للمعاملات في سلسلة SQL، ويقوم هذا الإجراء باسترجاع مجموعة مفيدة من المعلومات أيضاً.

تعليمات برمجة ١٥-١٥

```
Private Sub MovieInsert(iMovieId As Integer, _
   StrMovieName As String, _iLength As Integer, _
   StrRating As String, iYear As Integer)
'  Insert a new movie into the database

   qyMovieInsert(0) = 0
   qyMovieInsert(1) = strMovieName
   qyMovieInsert(2) = iLength
   qyMovieInsert(3) = strRating
   qyMovieInsert(4) = iYear

   qyMovieInsert.Execute
   MsgBox "The movie_id used was " & qyMovieInsert(0)
End Sub
```

تحذير مع إصدار RDO 2.0 الذي أتى مع الفيجوال بيسك ٥، لم يكن من الممكن تضمين مخطط قاعدة بيانات قبل أسماء الإجراء. وتقوم مكتبة RDO 2.0 التي تم مراجعتها والتي جاءت مع الإصدار السادس لفيجوال بيسك بحل هذه المشكلة مع أنها جاءت على حساب عدم القدرة في استمرار العمل بالحزم.

قطعة رموز PL/SQL المجهولة

لقد واجهنا هذا الكائن الغريب في الفصل التاسع، مقدمة إلى PL/SQL. وتعليمات برمجة PL/SQL المجهولة هي ببساطة مجموعة غير مسماة من أوامر PL/SQL بهيكل القطعة تم تحديدها بواسطة BEGIN والكلمة الأساسية END. وتم اعتبارها غير مسماة مقارنة مع الإجراءات والوظائف المخزنة المسماة. يمكنك تنفيذ تعليمات برمجة مجهولة من خلال ODBC وهذا يعني كيف نقوم بتنفيذ الإجراءات المخزنة.

وإذا قمنا بتطوير المفهوم خطوة للأمام أكثر، نجد أنه يمكن استخدام تعليمات البرمجة الموجودة داخل القطعة لأكثر من استدعاء جملة SQL. والمثال هو تضمين جملة COMMIT بعد جملة DML مثل DELETE.

```
StrSQL = "BEGIN DELETE FROM movies WHERE id = 123;"
StrSQL = strSQL & "COMMIT; "
StrSQL = strSQL & "END; "
```

قد توجد دواعي عديدة عندما يمكنك استخدام هذه الطريقة المحددة لأن RDO تحتوي على تثبيت آلي على كل إجراء بقاعدة البيانات ما لم تستخدم التحكم في الحركة BeginTrans/CommitTrans من الفيجوال بيسك.

استدعاء وظائف PL/SQL

لقد أغفلنا شيء هام وهو أن RDO يمكنها القيام بتنفيذ وظائف PL/SQL تماماً مثل الإجراءات المخزنة. فلا يمكنك تنفيذ وظيفة من خلال محرك JET لأنه لا يمكنك استرجاع المعاملات الخارجة، التي تعد الآلية الرئيسية للوظيفة. ولاستدعاء وظيفة، عليك استخدام نفس الصيغة التي شاهدناها للإجراءات المخزنة، ماعدا الآن فنقوم بتضمين قيمة الرجوع في سلسلة التحديد. لابد أن نتأكد أيضاً من تحديد اتجاه المعامل الأول هذا كـ rdParamReturnValue. توضح التعليمات البرمجية ١٥-١٦ وظيفة PL/SQL بسيطة، بينما توضح التعليمات البرمجية ١٥-١٧ تعليمات برمجة فيجوال بيسك التي تقوم باستدعاء الوظيفة وعرض قيمة الرجوع.

تعليمات برمجة ١٥-١٦

```
CREATE OR REPLACE FUNCTION GET_MOVIE_ID
( p_movie_name IN VARCHAR2 ) RETURN NUMBER
IS
/* Function to return the id for a movie name */
v_movie_id   number(6);
BEGIN
  SELECT movie_id INTO v_movie_id video_store.movies
     WHERE movie_name = p_movie_name;
  RETURN v_movie_id;
```

```
EXCEPTION
  WHEN OTHERS THEN
    RETURN -1;
END;
```

تعليمات برمجة ١٥-١٧

```
Dim qy1 As rdoQuery
Dim strSQL as String
strSQL = "{ ? = call video_store.get_movie_id (?) }"
Set qy1 = cn.CreateQuery("", strSQL)
qy1.rdoParameters(0).Direction = rdParamReturnValue
qy1(1) = "Awakenings"
qy1.Execute
MsgBox "The id for the movie is " & qy1(0)
```

العملية غير المتزامنة

تعد الإمكانات التي تجلبها RDO لمطوري الفيجوال بيسك قوية للغاية. وبإعطاء قيود العمل مع ODBC، من الصعب تخيل التطورات العديدة التي كان من الممكن تضمينها إلى RDO. ومع القدرة على استرجاع المعاملات الخارجة من الإجراءات المخزنة، تعتبر القدرة على تنفيذ عمليات محددة وغير متزامنة واحدة من أهم الخصائص المثيرة. وتوجد هذه الخصائص بوفرة في RDO 2.0.

ومع DAO/JET، إذا قمت بإجراء أي شئ يستخدم أوراكل، سواء كان يقوم بتنفيذ إجراء مخزن أو تشغيل استعلام، فسوف يتم إقفال التطبيق حتى يصبح الإجراء كامل. وهذا لا يعني إقفال تطبيقات أخرى، ولكن التطبيق المصمم جيداً من وجهة نظر المستخدم لن تستغرق مدة إقفاله أكثر من ثانية.

ومدلول العملية الغير متزامنة أنك تقوم ببدء تشغيل قاعدة البيانات ثم تستمر في المعالجة الداخلية، وعليك دورياً التدقيق لمشاهدة ما إذا كان تم إكمال هذه العملية أو، بالأحرى، ما زال العمل جاري وفق الحدث الذي يدل على أن العملية تم إكمالها. ولقد هذا الحل متاح في الفيجوال بيسك ٥ و RDO 2.0. يمكنك استخدام العملية غير المتزامنة باستخدام الطرق التالية:

- .OpenConnection

- OpenResultset.
- Requery.
- MoveLast.
- Execute.

في كل حالة، تقوم بوضع الثابت rdAsyncEnable في معامل الخيـــارات. توضـــح التعليمـــات البرمجية ١٥-١٨ مثال حول كيفية إجراء هذا.

تعليمات برمجة ١٥-١٨

```
Private Sub RefreshRecordset()
' Ensure the recordset is up to date
' If the selection box is not set to "ALL" then add a WHERE clause
    Dim strSQL As String

    strSQL = "SELECT movie_name, length, rating, year_of_release, "
& _
        "movie_id FROM video_store.movies ORDER BY movie_id"
    Set rsMovies = cn.OpenResultset(strSQL, rdOpenKeyset, _
            rdConcurRowVer, rdAsyncEnable)

End Sub
```

فعلى استدعاء الإجراء ShowRecord الانفراد عن هذا الإجراء وأن ينفذ عند إكمال العملية.

التأكد من الإكمال

طريقة واحدة لمعرفة وقت إكمال العملية غير المتزامنة هي فحـــص خاصيـــة StillExecuting للوحدة. ويمكنك تنفيذ ذلك في حلقة باستخدام جملة DoEvents في المنتصـــف، ولكـــن هـــذا لا يساعدك كثيراً في مرحلة التنفيذ. ففي الحقيقة قد يتسبب التدقيق الزائد في بطأ كل العملية. ومـــع هذا، يمكنك فعل أشياء أخرى في البرنامج الخاص بك على الأقل مثل تجهيز نمـــوذج. وكمثـــال لترميز جملة DoEvents هو

```
While cn.StillExecuting
   DoEvents
```

```
Wend
```

والطريقة الأفضل لتأكد من الإكمال هي وضع الاختبار في حدث Timer كما هو موضح في التعليمات البرمجية ١٥-١٩. ومع هذه الطريقة لابد أن تكون منتبه لتذكر تعطيل عداد الوقت في وقت التصميم وتشغيله مباشرة بعد استخدام طريقة CreateResultset.

تعليمات برمجة ١٥-١٩

```
Private Sub Timer1_Timer()
' This timer event is used to check for completion
' of the CreateResultset method
    If Not cn.StillExecuting Then
        Timer1.Enabled = False
        ShowRecord
    End If
End Sub
```

استخدام الأحداث

بالرغم من أنه يمكنك استخدام خاصية StillExecuting، تعرض RDO 2.0 تنفيذ معظم الأعمال لك بواسطة تزويد نموذج كامل يتم تشغيله بواسطة الحدث. وبواسطة تعريف الوحدة سواء كانت rdoConnection أو rdoResultset باستخدام الكلمة الأساسية WithEvents، يمكنك جعل التغييرات متاحة للتطبيق الخاص بك. توضح التعليمات البرمجية ١٥-٢٠ مثال حول كيفية استخدام الأحداث باستخدام rdoResultset التي شاهدناها مكونة بشكل غير متزامن في التعليمات البرمجية ١٥-١٨.

تعليمات برمجة ١٥-٢٠

```
Dim WithEvents rsMovies As rdoResultset
Private Sub cn_QueryComplete(ByVal Query As RDO.rdoQuery, _
        ByVal ErrorOccurred As Boolean)
' Process the QueryComplete event
    ShowRecord
End Sub
```

العملية الغير متزامنة في حيز التطبيق

عند استخدام خصائص RDO الغير متزامنة أو واحد من نماذج الوحدات الأخرى، عليك الانتباه إلى التصميم الخاص بك. ويمكنك الآن ترك المستخدمين في الاستمرار في عمل آخر أثناء حدوث إجراء قاعدة البيانات، ولكن عليك الانتباه إلى أن التطبيق لم يفقد ما يقوم به من إجراءات. والمستخدمون لا يريدون تنفيذ العملية ثم بعد دقائق تحصل على مربع رسالة يدل على العملية قد انتهت. ويعد هذا مفيد مع نوع الأنواع بالدفعة، ولكن بالنسبة للمهام الأصغر قد لا يكون مفيد بالمرة.

وللاستفادة العملية من العمليات الغير متزامنة، لابد أن يكون لديك شئ ما مفيد بحيث يتمكن التطبيق أو المستخدم من أداءه في وقت إجراء قاعدة البيانات.

وهناك قليل من الاقتراحات:

OpenConnection: على الرغم من أن وقت الاتصال الذي يستخدم RDO يفضل أن يكون قصير جداً، إلا أنه يمكنك استخدام وقت الاتصال لإتمام كل معالجات الإعداد المحلية غير المتزامنة المطلوبة. ويعد ذلك قيماً إذا أردت عمل أكثر من اتصال في نفس الوقت.

OpenResultSet: العملية المماثلة هي وضع طريقة OpenResultSet في حدث Form_Load أو، بالأحرى، الإجراء الذي تم استدعاءه من حدث Form_Load. ولن يقوم النموذج بالعرض حتى يتم ملء مجموعة النتائج بالكامل، ومن الممكن أن يقوم بالعرض بعد ثوان قليلة. ومع العملية غير المتزامنة، يمكنك عرض النموذج بسرعة بالغة، ولكن عليك تعطيل الوظائف المتعلقة بالبيانات حتى حدث Completion. قد يعطي هذا للتطبيق الخاص بك سرعة وهمية قد تكون أكثر من السرعة الحقيقية.

MoveLast: من الممكن أن يأخذ ملء rdoResultset مدة من الوقت جديرة بالملاحظة، خاصة إذا قمت باستخدام مكتبة خلية ODBC أبطأ. قد تستغرق هذه العملية ثانية أو ما إلى ذلك، ولكن حتى لو لم يكن لديك معالجة أخرى تقوم بها في الوقت الفاصل، فمن الممكن أن تجعل النموذج يظهر على الفور.

[illegible]

الترافقي بالصفوف الموجودة. وقد يستغرق هذا النوع من الاستيراد ١٠ دقائق، ولكن توجد بعض الحالات التي قد تستغرق ساعة أو أكثر ويجب تشغيلها بالدفعة في حالة كون النظام مفصول (يتم تشغيل النظام الليلي القديم). ومع هذا فإن الأيام التي تكون فيها قاعدة البيانات في حالة راحة عن العمل تصبح نادرة، واستخدام العملية الغير متزامنة يعني أن برنامج الفيجوال بيسك يمكنه التحكم في العمل بالحجم المتوسط بدون إقفال تطبيق المستخدم. وبالنسبة للأعمال التي تستغرق أكثر من دقائق قليلة وتلك الأعمال التي يتم تشغيلها بشكل منفرد، يجب وضعهم على خادم مصمم لهذا الغرض، وإلا أي إجراء آخر سوف يكون مخاطرة بفقد تتبع العمل إذا قام المستخدم بإنهاء التطبيق.

تحذير عليك الانتباه للعمليات غير المتزامنة لأنه من الممكن أن تؤدي إلى تأثيرات لا تتوقعها. على سبيل المثال، يظهر حدث cn_QueryComplete وقتما يقوم أي استعلام بالإكمال، وليس شرط أن يكون الاستعلام الغير متزامن. وإذا كان لديك استعلام فردي تم تنفيذه في الحدث، سوف تنتهي بحلقة بدون نهاية.

استرجاع جداول PL/SQL

واحدة من أحدث التطورات المتاحة لنا مع RDO (وكذلك مع وحدات بيانات ActiveX) هي القدرة على الوصول إلى جداول PL/SQL. ولقد شاهدنا هذه الجداول أولاً في الفصل العاشر، تكوين إجراءات مخزنة وتم اعتبارهم كمصفوفة حيوية قائمة على أساس الجدول. ومن الممكن الآن سحب هذه الجداول إلى مجموعة النتائج في ODBC على أساس النظام. وعند التطبيق تتحول الأكذوبة إلى حقيقة الأمر أن أوراكل لا يستطيع إرجاع rdoResultset من إجراء مخزن، فهذه ليست مهمة سهلة مثلما يكون الحال مع خادم SQL لمايكروسوفت؛ فالنظام يعتمد على مشغل ODBC المطلوب. (على سبيل المثال، تم طلب مشغل ODBC لمايكروسوفت لفيجوال بيسك ٥). ونظراً للتغييرات الثابتة في إمكانات مشغلات ODBC، قد تريد المحاولة مع جداول PL/SQL ولكن العلية أكثر ثقة مع ADO. وبإعطاء التوليفة الصحيحة من المكونات، يمكنك استرجاع PL/SQL باستخدام RDO، ولكن نترك المناقشة للفصل ١٧، وحدات بيانات ActiveX. وإذا كنت تريد محاولة استرجاع جداول PL/SQL عن طريق RDO، عليك مشاهدة إصدار ADO لأن الصيغة نفس الشيء.

معالجة الأخطاء

بينما تزود DAO بمجموعة Errors لأخطاء البيانات، فإن RDO تزود بمجموعة rdoErrors. ويمكنك استخدام هذه المجموعة بنفس طريقة استخدام مجموعة Errors. وعليك الرجوع إلى الفصل الثالث عشر، قضايا تصميم الفيجوال بيسك، لإلقاء نظرة عامة على معالجة الأخطاء في بيئة قاعدة البيانات.

الجدول ١٥-١٣: خصائص وحدة rdoError الخاصة بمجموعة RDO Errors.

الخاصية	البيان
Number	توضح هذه الخاصية رقم صحيح (رقم صحيح طويل)
Description	توضح هذه الخاصية سلسلة تحتوي على بيان الخطأ.
Source	توضح هذه الخاصية اسم الوحدة التي سببت الخطأ.
SQLRetcode	توضح هذه الخاصية تعليمات برمجة لإكمال الجملة المنفذة.
SQLState	تكشف هذه الخاصية عن تعليمات برمجة أخطاء ODBC الداخلية.
HelpContext	توضح هذه الخاصية معرف سياق ملف Help لرقم الخطأ.
HelpFile	توضح هذه الخاصية اسم الملف بالكامل لملف Help للفيجوال بيسك.

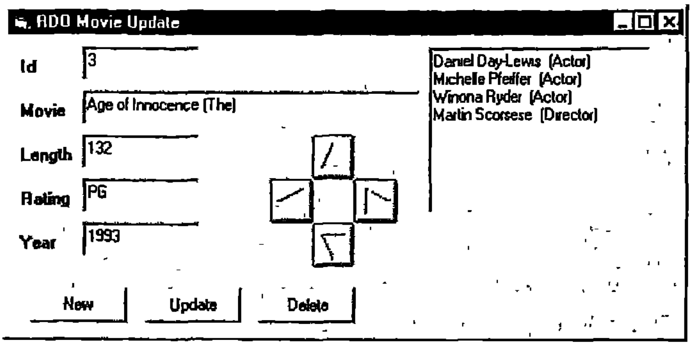
التجميع كلية

مع الانتقال إلى RDO، من المفضل أن تقوم بالانتقال من مجموعات النتائج القابلة للتحديث إلى استخدام مجموعات النتائج الخاصة بالقراءة فقط عن طريق استدعاءات الإجراء المخزنة لكي تقوم بالتحديث. وإمكانية أخرى وهي القيام بإنشاء مكونات الطبقة الوسطى لإجراء التحديث عن طريق أوامر SQL (وفي هذه الطريقة، يمكنك إعطاء الامتيازات INSERT وupdate وdelete للمكونات التي يتم التحكم فيها باهتمام). وبنسبة لسياق هذا القسم، سوف نتعرض لعملية استخدام الإجراءات المخزنة، لأنها تقوم بتسهيل أصعب الطرق المعقدة. وعلى كل، من الممكن بناء كلا مجموعات النتائج القابلة للتحديث وأوامر SQL في طريقة سهلة بمجرد تغطية المبادئ.

وسوف ننتهز الفرصة للتطوير على مثال DAO بواسطة عرض المشتركين في الفيلم الذي تتم إعادة عرضه. يمكنك مشاهدة النموذج في الشكل ١٥-٤.

الشكل ١٥-٤

نموذج تحديث فيلم RDO



نجد نفس تعليمات البرمجة متطابقة كما شاهدنا في الفصل ١٤، وحدات الوصول للبيانات، ولكنه يحتوي على الطرق التي قمنا بتغطيتها في هذا الفصل. ونقطة واحدة معينة عليك ملاحظتها هي الاستعلام الذي يقوم بسحب المشتركين إلى الخارج (الممثلون والمخرجون) المشتركين في الفيلم المعروض. وهذا من الشيق لن الاستعلام أكثر تعقيداً أكثر مما شاهدناه على مدار هذا الفصل حتى الآن. ويعتبر تكوين استعلام لمعالجة العملية طريقة ممتازة لإخفاء تعقيد جملة SQL. توضح التعليمات البرمجية ١٥-١٢ تعليمات برمجة إجراء CreateQueries، الذي يقوم أولاً بإعداد استعلام Movies_Insert3 ثم إعداد rdoQuery لاسترجاع قائمة المشتركين في الفيلم. تحتوي هذه القائمة على معامل واحد لـ Movie_Name كما يحتوي على وظيفة DECODE التي تقوم بتحويل تعليمات البرمجة المشتركة لاسم أكثر معنى. ولمزيد من المعلومات حول تكوين استعلامات SQL، راجع الفصل ٧، الاستعلامات.

تعليمات برمجة ١٥-٢١

```
Private Sub CreateQueries()
'  Set up any queries required in this module
   Dim strSQL As String
   On Error GoTo ErrorHandler

'  Set up the query to call the insert stored procedure.
   strSQL = "{call video_store.movies_insert3 (?, ?, ?, ?, ?) }"
   Set qyMovieInsert = cn.CreateQuery("", strSQL)
```

```
' Set up the query to select the participants.
  strSQL = "SELECT first_name || ' ' || last_name full_name,"
  strSQL = strSQL & " DECODE (role, 'A', 'Actor', 'D', 'Director') _
         & "role, movie_id"
  strSQL = strSQL & " FROM video_store.participants, " & _
              video_store.movie_participants"
  strSQL = strSQL & " WHERE movie_id = ? "
  strSQL = strSQL & "  AND movie_participants.participant_id =" & _

     " participants.participant_id"
  strSQL = strSQL & "  ORDER BY role"

  Set qyParticipants = cn.CreateQuery("", strSQL)

  Exit Sub
ErrorHandler:
  If Err = 3146 Then
    MsgBox "Error creating query : " & rdoErrors(0).Description
  Else
    MsgBox "Error creating query : " & Err.Description
  End If

End Sub
```

عرض هيكل قاعدة البيانات

كما شاهدنا من خلال DAO، يمكنك عرض هيكل أي جداول لك حق الوصول إليها. والاختلاف الرئيسي بين DAO وRDO هو أنه لا يمكنك تغيير هيكل قاعدة البيانات باستخدام أوامر RDO. حتى DAO غير موصى بها لهذا الغرض، وسوف يكون مسئول قاعدة البيانات في راحة في كل مكان بأن المستخدمون ليس في حاجة الآن لتغيير هيكل قاعدة البيانات.

rdoTables

تحتوي مجموعة rdoTables على وحدة rdoTable فقط لكل جدول بأوراكل لك حق الوصول إليه. وهذا لا يدل على أنه يمكنك تحديث تلك الجداول، ولكن يتيح لك النظر إلى تلك هياكل تلك الجداول. وأنواع الأشياء التي يمكنك أدائها باستخدام rdoTable هي نفس الأشياء التي فعلناها باستخدام TableDefs في DAO، لذا سوف أقوم بعرض نظرة عامة موجزة حول هيكل الوحدات.

ويمكنك عرض أي الجداول لأوراكل التي لها حق الوصول إليها باستخدام التعليمات البرمجة ١٥-٢٢، الذي يدور على كل وحدات rdoTable.

تعليمات برمجة ١٥-٢٢

```
Dim tbl As rdoTable
   For Each tbl In cnMovies.rdoTables
      Debug.Print tbl.Name
   Next
```

RdoColumns

تحتوي DAO على وحدات Field بينما تحتوي RDO على وحدات rdoColumn. والخصائص الهامة لوحدة Field موضحة في الجدول ١٥-١٤.

الجدول ١٥-١٤: خصائص وحدة rdoColumns التي تتطابق مع أوراكل.

الخاصية	البيان
AllowZeroLength	تدل هذه الخاصية على أن NULLs مسموح بها في العمود.
Attributes	تقوم هذه الخاصية بإرجاع قائمة السمات الخاصة بالعمود. ويعد هذا مفيد أساساً لمستخدمي خادم SQL.
BatchConflictValue	تقوم هذه الخاصية بإرجاع قيمة قاعدة البيانات الحالية (استخدم هذه الخاصية عند استخدام تحديثات الدفعة).

الجدول ١٥-١٤: خصائص وحدة rdoColumns التي تتطابق مع أوراكل.

الخاصية	البيان
ChunkRequired	تدل هذه الخاصية على أنه عليك استخدام طريقتي GetChunk وAppendChunk.
KeyColumn	تدل هذه الخاصية على ما إذا كان هذا المفتاح فريد.
Name	توضح هذه الخاصية اسم الحقل
OrdinalPosition	توضح هذه الخاصية موضع العمود في مجموعة rdoColumns.
OriginalValue	توضح هذه الخاصية قيمة العمود عند وقت ملء مجموعة النتائج في المرة الأولى.
Required	تدل هذه الخاصية على ما إذا كان قد تم وضع قيد NOT NULL.
Size	توضح هذه الخاصية حجم الحقل بالبايت.
SourceColumn	توضح هذه الخاصية اسم العمود من جدول المصدر.
SourceTable	توضح هذه الخاصية اسم الجدول المرفق (إذا وجد)
Status	تدل هذه الخاصية على ما إذا كان تم تغيير قيمة العمود.
Type	توضح هذه الخاصية الرقم الذي يمثل نوع العمود الذي تم تفسيره بواسطة RDO. وتحتاج هذه الخاصية إلى تفسير قبل أن تكون صالحة مقابل قاعدة بيانات أوراكل.
Updatable	تدل هذه العملية على ما إذا كان من الممكن تحديث العمود.
Value	تشير هذه الخاصية إلى الخاصية الافتراضية ولكن تكون متاحة للاستخدام فقط عند الإشارة إلى مجموعة النتائج.

يمكنك سرد خصائص العمود بسهولة بالغة؛ على سبيل المثال، تدرج تعليمات البر مجة العينة في التعليمات البرمجية ١٥-٢٣ خاصية Type. إذا كنت تريد تفسير نوع العمود باصطلاح أوراكل، استخدم الوظيفة الموضحة في التعليمات البرمجية ١٥-٢٤.

تعليمات برمجة ١٥-٢٣

```
For Each tbl In cn.rdoTables
   Debug.Print ""
   Debug.Print tbl.Name
   For Each col In tbl.rdoColumns
      Debug.Print col.Name, FieldType(col.Type)
   Next
Next
```

تعليمات برمجة ١٥-٢٤

```
Private Function FieldType(intType As Integer) As String
' Convert the rdoColumn type to an Oracle column type
   Select Case intType
      Case rdTypeINTEGER
         FieldType = "NUMBER"
      Case rdTypeDECIMAL
         FieldType = "NUMBER"
      Case rdTypeNUMERIC
FieldType = "NUMBER"
Case rdTypeREAL
         FieldType = "NUMBER"
      Case rdTypeDOUBLE
         FieldType = "NUMBER"
      Case rdTypeDATE
         FieldType = "DATE"
      Case rdTypeCHAR
         FieldType = "CHAR2"
      Case rdTypeVARCHAR
         FieldType = "VARCHAR2"
      Case rdTypeLONGVARBINARY
         FieldType = "RAW"
      Case rdTypeTIMESTAMP
```

```
        FieldType = "DATE"
    Case Else
        FieldType = "Unknown"
  End Select
End Function
```

الفهارس

لا تحتفظ RDO بمجموعة منفصلة من الفهارس. ومع هذا، فلا يعد ذلك تعقيد لأنه يمكنك دائما استخدام خاصية KeyColumn لوحدة rdoColumn.

عيوب RDO

تعتبر RDO تقنية القمة للوصول إلى البيانات من مصادر بيانات ODBC. ولقد تم تصميمها حول API الخاصة بـ ODBC، التي لا تعتبر ميزة تامة مع أوراكل. وعلى الرغم من وجود تطويرات هائلة جارية على DAO، إلا أن بعض الخصائص غير متاحة لمستخدمي أوراكل لأن بنية أوراكل تختلف تماما عن خادم SQL. وكما شاهدت في هذا الفصل، كل مجهود يبذل لإظهار أوراكل في الشكل التام، إلا أنه ما زالت بعض العيوب في RDO التي يجب علينا ملاحظتها:

- لا يمكنك استرجاع متغيرات خلية PL/SQL من الإجراءات المخزنة.
- تعتبر RDO تقنية قديمة لمايكروسوفت ولن تقبل أي تطوير، ولكن يمكن معالجة الأخطاء الصغيرة فقط.
- تعتمد RDO على مشغلات ODBC ومشاكل التضارب الناتجة عنها.
- لا تقوم RDO بدعم معاملات الربط لأوراكل.
- لا تعتبر RDO بنفس سرعة ADO، خاصة عند استخدام مزود OLE-DB.

مستقبل RDO

يوجد بعض التأمل حول مستقبل RDO الآن خاصة وأن وحدات البيانات النشطة (ADO) تأتي إلى الظهور. فأين إذا يمكنك وضع مجهود التطوير؟ والحقيقة هي أن على الرغم أن ADO سوف توجد بها ميزات لن توجد في RDO، فإن ADO من الممكن لن يكون لديها إمكانات RDO. لمدى محدد، كلتا الوحدتان تكمل إحداهما الأخرى، وسوف يكون وقت طويل قبل أن يتم تـرك RDO بواسطة المطورين.

سوف تقوم مايكروسوفت بدعم RDO لبعض الوقت، على الأقــل طالمــا يقــوم المطوريــن باستخدامها بشكل نشط، ومع هذا فأي تطويرات سوف تكون قليلة جداً. وإذا كان لديك اســتثمار بديل في تطبيقات ADO، لابد أن تكون حذر عند نقلهم إلى ADO، على الأقــل حتــى يتــهيأ لا الحصول على مجموعة معالجات التحويل. ولكل أعمال التطوير الجديدة المتوقعة، لابد أن تضــع في اعتبارك جدياً الانتقال إلى ADO.

خُلاصة

لقد قمنا في هذا الفصل بتغطية RDO بتفاصيل كافية لعمل وصــلات بقــاعدة بيانــات أوراكــل واسترجاع مجموعات النتائج من جداول أوراكل وكذلك استخدام وحدة rdoConnection لتنفيــذ أوامر SQL واستدعاء الإجراءات المخزنة. ولقد شاهدنا الخواص التي يوجد بها مكسب فعاليــة بديل مقارنة باستخدام محرك JET في DAO ومنها القدرة على إرجاع المعاملات الخارجة مــن الإجراءات المخزنة وكذلك تكوين rdoResultsets من الإجراءات المخزنة التي تقوم بإرجــاع جداول PL/SQL. وتم أيضاً مناقشة العملية غير المتزامنة مع المميزات الخاصــة بــها. وبعــد التعرض لمعالجة الأخطاء باستخدام RDO، ولقد شاهدنا مثال حول وضع هذه الأساليب معاً.

وحتى وصول ADO، كانت RDO الأحدث فيما يخص الوصول إلى البيانات. وبسبب قضايــا الترخيص. تم إتاحة بعض الميزات إلــى بيئــات أخــرى مثــل مايكروســوفت إكســل عــبر ODBCDirect. الذي سوف نتعرض له الآن بشكل موجز.

١٦

ODBCDirect

ODBCDirect هو اسم تم إعطاؤه لجزء من DAO 3.51، وهذا الجزء يعطيك صلاحية الوصول إلى بعض خصائص وحدات البيانات البعيدة التي قمنا بتغطيتها في الفصل ١٥. ويمكن وصفها أنها تقع في منطقة وسط بين DAO/JET وRDO لأن لديها نفس نموذج الوحدة لـ DAO الأصلي ولكن بأنواع امتداد RDO. يقوم فجوال بيسك بتحميل مكتبة RDO 2.0 لـ ODBCDirect بدلا من محرك JET، لذا لا يمكنك الوصول إلى الوظيفة الكاملة لـ RDO 2.0. ونظرا لهذه القضايا، فإنه من الصعب وضع ODBCDirect في فئة، ولكن هذا الأمر لن يؤثر في استفادتك التامة منه.

ODBCDirect كانت أساسا جزءا من DAO 3.1 كوظيفة إضافية يمكن تحميلها. وأصبحت متاحة تماما مع نموذج 32-bit، ولهذا سيكون من الصعب على هؤلاء الذين ما زالوا يستخدمون 16-bit أن يجروا تحديثا من بيئة JET.

ODBCDirect يمكنها فعل نفس الأشياء التى تقوم بها DAO/JET لقواعد البيانات البعيدة ولكن من غير أن تترك أثر واضح. وبالرغم من عدم استخدامها لربط قواعد البيانات متغايرة الخصائص، فإنك يمكنك التحكم في الأسهم بشكل أفضل، بالإضافة إلى الوصول إلى معاملات إخراج الإجراءات المخزنة والاستعلامات والقدرات غير المتزامنة والتحكم في استخدام الاتصال.

ملاحظة إذا أردت ربط قواعد بيانات مختلفة مع بعضها البعض، يمكنك شراء أحد منتجات جيت واي لأوراكل، وهذا المنتج يجعل منتجات قواعد البيانات الأخرى يمكن الوصول إليها بنفس سهولة الوصول إلى أوراكل.

أثارت ODBCDirect الاهتمام عند أول ظهور لها في عام ١٩٩٦. RDO كانت متاحة حتى VB5 في نسخة المؤسسة فقط، ولكن مع VB6، فهي متاحة الآن في النسخة المهنية. ODBCDirect متاحة الآن في النسخة المهنية وكذلك متاحة لبيئات تطبيقات فجوال بيسك أخرى، مثل اكسل. في الحقيقة، هذا هو السوق الذي استهدفته أساسا. وعلى الرغم من قيام ADO بالتحسين في كل وظائف ODBCDirect وستقوم بحل محلها في أغلب الحالات، ما زالت ODBCDirect تقوم بإمداد تحديث نسبيا من DAO بالمقارنة مع RDO أو ADO.

والمواضيع التي سيغطيها هذا الفصل هي:

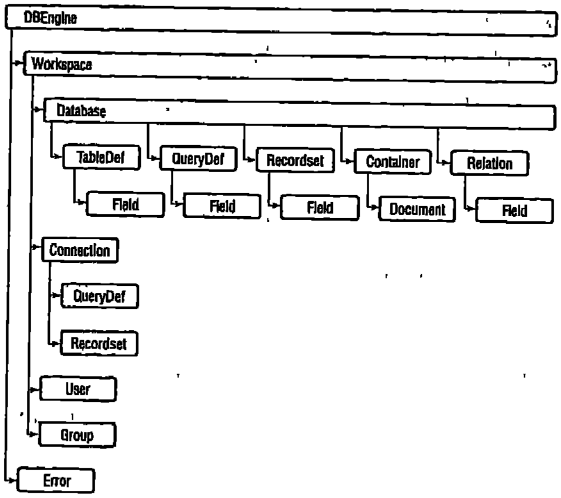
- عرض تدرج وحدة ODBCDirect.
- الوصول إلى ODBCDirect.
- تكوين مجموعات السجلات.
- استخدام QueryDefs.

- استدعاء الإجراءات المخزنة.
- سلبيات ODBCDirect.

ملاحظة: معظم الموضوعات التي تم تغطيتها هنا مطابقة لطرق DAO/JET أو طرق RDO. ولهذا لن نقوم بتغطيتها بالتفصيل، ولكن سنحاول إلقاء الضوء على النقاط والاختلافات الواضحة.

تدرج وحدة ODBCDirect

تدرج وحدة ODBCDirect تشبه إلى حد بعيد تدرج DAO الذي تعرفنا عليه في الفصل ١٤ "وحدات الوصول إلى البيانات". على الرغم من ذلك، كما ترى في الشكل ١٦-١، توجد إضافة مجموعات ووحدات قليلة. وبالأخص، يتضمن النموذج الجديد وحدات ومجموعات Connection وQueryDef وParameters. وسنقوم بتقديم تلك العناصر الجديدة في هذا المقطع وسنقوم بتفصيلهم خلال الفصل.



الشكل ١٦-١ تدرج وحدة ODBCDirect

DBEngine

وحدة المستوى العلوي في تدرج وحدة ODBCDirect هي DBEngine، كما كانت مع تدرج DAO/JET. عند استخدام خيار ODBCDirect، ستستخدم خصائص DBEngine أكثر من استخدامك للخصائص عند التعامل من خلال JET. وكما كان الحال مسبقا، فإن وحدة DBEngine تحتوي على مجموعات أخطاء ومساحات عمل.

Workspace

أصبحت مجموعة مساحات العمل الآن بناء تنظيمي أكثر أهمية لأنها الطريقة الوحيدة التي يمكنك من خلالها التحكم في أي أنواع البيئات تريد التعامل معها: JET أو ODBCDirect. كما تعطيك القدرة على التعامل مع كلتا البيئتين في نفس الوقت. وبدلا من احتواء مجموعة Databases فقط، فإنها تتضمن الآن وحدة Connection، التي تمد أغلب الخصائص الجديدة. وتحت ODBCDirect، تحتوي وحدة Workspace على مجموعات Databases وConnections.

Database

وحدة Database هي أهم وحدة لـ JET، ولكن لتستفيد من استخدام ODBCDirect أقصى استفادة، ستضطر إلى البعد عنها إلى وحدة Connection. على الرغم من ذلك، في الوقت الحالي، يمكنك استخدام وحدة Database بنفس الطريقة التي قمنا بها من قبل مع JET. في الحقيقة، جعل فريق ODBCDirect وحدة الوصلة مستخدمة بواسطة وحدة قاعدة البيانات المتاحة للتطبيق، ولكن لن يساعدك هذا على الاستغلال الأمثل للوصلة التي ينشئها التطبيق.

الوصلة

بواسطة إعداد مساحة العمل مع الخيارات الصحيحة، فأنت تجعل وحدة Connection متاحة للتطبيق. وهذه هي الوحدة التي ستعطيك القدرة على اختيار الخلايا، واستخدام التقنيات غير المتزامنة واستخدام QueryDefs (المعادل لوحدات rdoQuery التابعة لـ RDO).

الوصول إلى ODBCDirect

كما رأينا في الفصل ١٤، يجب جعل مكتبة DAO متاحة للتطبيق بواسطة تعليمها في خيار References الموجود في قائمة Project في فجوال بيسك. وإذا كنت تستخدم مساحة العمل الافتراضية، سيكون لديك محرك JET، وسيكون أول شيء علينا الوصول إليه هو مساحة عمل جديدة. ويمكننا القيام بهذه المهمة بواسطة طريقة CreateWorkspace لوحدة DBEngine والتي لها الصياغة التالية:

```
Set workspace = CreateWorkspace(name, user, password, type)
```

وتحتوي هذه الطريقة على المعاملات التالية:

name "الاسم": اسم مساحة العمل. يمكنك استخدام سلسلة خالية للاسم إذا كنت ستقوم فقط بتكوين مساحة عمل واحدة جديدة.

user "المستخدم": المستخدم الافتراضي لمساحة العمل. يتم استخدام هذا الاسم إذا كانت سلسلة الوصلة لا تقوم بإمداد قيمة UID.

password "كلمة السر": كلمة السر لاسم المستخدم الافتراضي. يتم استخدام كلمة السر هذه إذا لم تقم بتضمين حقل PWD في سلسلة الوصلة.

Type "نوع": نوع مساحة العمل التي ترغب في تكوينها. والخيارت موضحة في الجدول ١٦-١، وكما ترى، يجب استخدام dbUseODBC للوصول إلى نموذج ODBCDirect.

الجدول ١٦-١: أنواع مساحة عمل DAO التي يمكنك تكوينها

الثابت	الوصف
DbUsejet	يقوم هذا الثابت بتكوين Microsoft Jet Workspace.
DbUseODBC	يقوم هذا الثابت بتكوين ODBCDirect Workspace.

وقد تم إضافة نوع المعامل خصيصا لـ ODBCDirect. ونوع المعامل اختياري، وإذا تركتها، سيكون المعامل الافتراضي هو JET Workspace. وتوضح التعليمات البرمجية ١٦-١ طريقة Create Workspace.

تعليمات برمجية ١٦-١

```
Private wks As Workspace
Private Sub ConnectToDatabase()
'  Connect to the Oracle database via ODBCDirect
Set wks = CreateWorkspace("WKS", "", "", dbUseODBC)
End Sub
```

ويمكن تكوين ODBCDirect Workspace بطريقة أخرى قد تجدها أسهل. تتحكم خاصية DefaultType لوحدة DBEngine في أنواع مساحة العمل التي يتم تكوينها وذلك بعد إعداد هذه

الخاصية. إذا قمت بالإعداد قبل الإشارة إلى Workspace(0)، سيكون الافتراض ODBCDirect Workspace في أول مرة تشير إليها. وفي هذه الحالة، يمكنك تجنب طريقة CreateWorkspace بأكملها. يقوم السطر التالي بتوضيح هذا.

```
DBEngine.DefaultType = dbUseODBC
```

ولأن هذه هي أسهل وسيلة للانتقال من بيئة JET، سنستخدم نفس الوسيلة لتكوين مساحة عمل من الآن فصاعدا. ويجب قبل ترك وحدة Worksapce إعداد خاصية DefaultCursorDriver كما فعلنا مع RDO. ويجب القيام بهذا قبل تكوين أي وصلات. وتم توضيح مشغلات الخلايا الممكنة في الجدول ١٦-٢.

الجدول ١٦-٢: القيم الممكنة لـ DefaultCursorDriver.

الثابت	الوصف
DbUseDefaultCursor "الافتراضي"	لقواعد بيانات أوراكل، يكون افتراض هذا الثابت مكتبة خلية ODBC.
DbUseODBCCursor	يستخدم هذا الثابت مكتبة خلية ODBC. ويفضل استخدام مكتبة خلية مجموعة العميل.
DbUseServerCursor	يستخدم هذا الثابت خلايا برامج الخادم.
DbUseClientBatchCursor	يستخدم هذا الثابت مكتبة خلية مجموعة العميل. وهذا الخيار هو أسرع مشغل لخلية الإزاحة ويفضل استخدامه لتحديثات المجوعات.
DbUseNoCursor	يستخدم هذا الثابت خلايا للأمام فقط وللقراءة فقط.

ملاحظة

يجب عدم الخلط بين خيارات DefaultCursorDriver هذه وبين أنواع الخلايا نفسها. فمشغلات الخلايا هي فقط المكتبة التي تتيح الأنواع المختلفة من الخلايا. والهدف الأساسي من تحديد هذا الخيار هو اختيار مكتبة خلية أكثر كفاءة لأي حالات تواجهنا. ولا يزال يجب علينا تحديد الخلية عند تكوين مجموعة السجلات "Recordset".

ومما سبق سنحاول إنتاج إجراء سيعطينا البيئة الصحيحة لاستخدام ODBCDirect. وتوضح التعليمات البرمجية ١٦-٢ كيفية القيام بهذا الإجراء. ويبقى فقط تكوين وصلة، وهذا ما سنتعرف عليه في الجزء التالي.

تعليمات برمجة ١٦-٢

```
Private Sub ConnectToDatabase()
   Connect to the Oracle database via ODBCDirect
   DBEngine.DefaultType = dbUseODBC
   Workspaces(0).DefaultCursorDriver = dbUseClientBatchCursor
End Sub
```

عمل الوصلة

وبعد وضع الأساس لبيئة الوصلة، علينا الآن تكوين الوصلة الفعلية (ويمكنك تكوين أكثر من وصلة إذا كنت تتعامل مع قواعد بيانات متعددة). وعليك تحديد أيهما ستستخدم. فيمكنك الاستمرار مع تنسيق DAO/JET وتستخدم وحدة Database بنفس الطريقة التي رأيناها في الفصل ١٤، أو يمكنك انتهاج الطريقة الجديدة، من خلال وحدة Connection. وتعتبر وحدة Connection أكثر ملائمة مع قواعد البيانات البعيدة، مثل أوراكل، وهذا هو الطريق الذي سنتبعه حتى انتهاء هذا الفصل. وبدونه، لن تستطيع استخدام عمليات QueryDefs أو العمليات غير المتزامنة.

ووحدة Connection هي المعادل الأساسي لوحدة Database التي استخدمناها في DAO/JET. وتقوم JET بتكوين ومعالجة أي وصلات تحتاجها، وتساعدك ODBCDirect على تحكم أكثر في التطبيق بواسطة كشفها للوصلات. وهذه تعتبر ميزة كبيرة، فبالرغم من أن JET تحاول استخدام الوصلات بأكثر كفاءة ممكنة، فإنا لا تستوعب بشكل كامل قدرات قواعد بيانات Back-end التي قد تتعامل معها. والنتيجة أنه يمكننا عمل أي شيء من خلال وصلة لكل قاعدة بيانات، في الوقت الذي تحتاج فيه JET إلى وصلتين على الأقل. والسبب الوحيد لعدم استخدام وحدة Connection هو أنه في حالة ما إذا أردت كتابة تعليمات برمجة يمكنك استخدامه للتبديل للخلف أو للأمام بين أوراكل وJET. هذا ليس غير مستحب.

أكسس وأوراكل

يوجد جانبية لاستخدام قاعدة بيانات مايكروسوفت اكسس لتطبيقات النماذج الأولية. وهذا منطقي لأنه يمكنك تكوين قاعدة البيانات من فجوال بيسك باستخدام مدير بيانات فجوال (VisData). وهذا الأمر يكون مفيداً إذا لم يكن بعد لديك تركيب أوراكل أو لم تكن قد اتفقت مع أي مورد قاعدة بيانات. ويكون هذا الميل لاستخدام إكسس عند استخدامك ODBCDirect لأنها تعطيك القدرة على كتابة تطبيق يقوم بمعالجة كلتا البيئتين، وربما فحص أي نوع من مساحة العمل أنت فيه قبل تنفيذ أي أمر.

على الرغم من ذلك، فنظرا للاختلاف الكبير بين صياغة JET وأوراكل، فإنك قد يكون عليك إعادة كتابة كمية كبيرة من الرموز. ويجب على الأقل استخدام مشغل ODBC لاكسس إذا كنت مصر على انتهاج هذا الطريق. ولكن يفضل تجربة استخدام أوراكل ثم استخدام هذه الطريقة مع مرحلة النماذج الأولية. إذا قمت باستخدام أدوات تكوين الجداول المتاحة في فجوال بيسك٦، ستحصل لفترة على القليل من خبرة أوراكل.

ولتكوين وصلة، قم باستخدام طريقة OpenConnection، صياغتها على النحو التالي:

```
Set connection = workspace.OpenConnection (name, options, readonly,
connect)
```

والمعاملات لطريقة OpenConnection هي:

name "الاسم": تستخدم هذا المعامل لإمداد اسم بيانات المصدر المسجلة (DSN) للوصلة، أو إذا قمت بإمداد DSN في سلسلة الوصلة، يمكنك إمداد اسم عشوائي للوصلة.

Options "الخيارات": هذا المعامل هو توليفة من معامل الحث RDO (لتحديد إمكانية وكيفية حث مدير المشغل ODBC لمعلومات الوصلة) وبين المعامل لتحديد العملية غير المتزامنة. والقيم الممكنة موضحة في الجدول ١٦-٣. ويفضل استخدام dbDriverNoPrompt لكي لا يكون هناك فرصة للمستخدم لرؤية توجيهات مدير المشغل.

Readonly (للقراءة فقط) يجب إعداد هذا المعامل إلى False إذا كنت تنوي استخدام مجموعات نتائج يمكن تحديثها في هذه الوصلة، وإلا، قم بإعداد هذا المعامل إلى True. ويدعم هذا المعامل تأمين أوراكل، ولا يقوم بحل محله.

Connect (الاتصال) يحمل هذا المعامل سلسلة وصلة ODBC التي رأيناها في الفصل ١١ "تقنية جانب المزود" إذا لم تقم بتضمين حقول UID أو PWD، ستأخذ الافتراضات من وحدة Workspace.

الجدول ١٦-٣: القيم الممكنة لمعامل الخيارات

الثابت	الوصف
DbDriverNoPrompt	يستخدم مدير مشغل ODBC سلسلة الوصلة المتوفرة في خصائص الوصلة والاسم. وستسبب المعلومات الغير كافية حدوث خطأ. ويكون هذا إلزاميا إذا كنت ترغب في التعامل مع خادم حركة مايكروسوفت.
DbDriverPrompt	يعرض مدير مشغل ODBC مربع حوار ODBC Data Sources، الذي يعرض أي معلومات متصلة متوفرة في معاملات الاسم أو الوصلة.
DbDriverComplete "الافتراض"	إذا تضمن معامل الوصلة كل المعلومات الضرورية لعمل وصلة، سيستخدم مدير مشغل ODBC هذا المعامل. وإلا، سيكون الحال كما كان عند تحديد dbDriverPrompt.
DbDriverCompleteRequired	يتصرف هذا الثابت مثل dbDriverComplete إلا أن مشغل ODBC يقوم بتعطيل التوجيهات لأي معلومات غير مطلوبة لإتمام الوصلة.
DbRunAsync	ينفذ هذا الثابت طريقة OpenConnection بطريقة غير متزامنة. يمكن إضافة هذا الثابت إلى ثوابت الخيارات الأخرى.

وكما رأينا مع RDO، يوجد مجموعة مركبة من الخيارات هنا، ولكن يمكننا تقليلهم إلى عدد أقل. تتضمن التعليمات البرمجية ١٦-٣ رموز لوحدات السيناريو الثلاث الممكنة: خلايا يمكن تحديثها وخلايا للقراءة فقط. ولا تقوم هذه الرموز بتكوين الخلايا، ولكن تحتاج إلى إعدادها بحيث يمكنك التوفيق بين كل أنواع مجموعات السجلات التي تريد إعدادها في الوصلة. بمعنى أنه إذا كنت تخطط لتكوين مجموعة سجلات يمكن تحديثها، سيكون عليك إعداد الوصلة.

تعليمات برمجة ١٦-٣

```
Private cn As Connection

Private Sub ConnectToDatabase()
'  Connect to the Oracle database via ODBCDirect
Dim strConnection As String

On Error GoTo ErrorHandler

DBEngine.DefaultType = dbUseODBC
strConnection =
"ODBC;DSN=movies;UID=video_user;PWD=ummagumma;"

'  Choose one of the following connections ......
'  Prepare connection for scrollable updatable cursors
Workspaces(0).DefaultCursorDriver = dbUseClientBatchCursor
Set cn = Workspaces(0).OpenConnection("", dbDriverNoPrompt, _
False, strConnection)
'  Prepare connection for scrollable read-only cursors
Workspaces(0).DefaultCursorDriver = dbUseClientBatchCursor
Set cn = Workspaces(0).OpenConnection("", dbDriverNoPrompt, _
True, strConnection)
'  Prepare connection for firehose cursors
Workspaces(0).DefaultCursorDriver = dbUseNoCursor
Set cn = Workspaces(0).OpenConnection("", dbDriverNoPrompt, _
True, strConnection)
Exit Sub
```

```
ErrorHandler:
If Err = 3146 Then
MsgBox Errors(0).Description
Else
MsgBox Err.Description
End If

End Sub
```

تكوين مجموعات السجلات

بمجرد إعداد البيئة مع مكتبات الخلايا الصحيحة والتأكد من استخدام ODBCDirect، يمكنك الآن الانتقال إلى تكوين مجموعة سجلات "Recordset". وهنا كذلك لديك العديد من الخيارات أكثر مما تحتاج، خاصة فيما يختص باختيار الخلايا. وصياغة طريقة OpenRecordset على النحو التالي:

```
Set recordset = _
  database_object.OpenRecordset (source, type, options, lockedits)
```

ولهذه الطريقة المعاملات التالية:

Source: وهذه سلسلة تقوم بتمثيل مصدر البيانات لمجموعة السجلات الجديدة. وستكون جملة SQL موضوعة بين علامات اقتباس مزدوجة، مثل "SELECT * FROM videos WHERE status='A'". لا تقوم بتضمين فاصلة منقوطة في آخر السطر.

Type: ثابت اختياري يشير إلى نوع مجموعة السجلات المراد فتحها، كما هو محدد في الجدول ١٦-٤.

Options: توليفة اختيارية من الثوابت التي تقوم بتحديد خصائص مجموعة السجلات الجديدة. يمكنك استخدام ثابت أو أكثر من الثوابت المحددة في الجدول ١٦-٥. وإذا قمت باستخدام أكثر من ثابت، يمكنك إضافتها مع بعضها. وتوجد ثوابت أكثر محددة في ملفات Help (التعليمات)، ولكن توجد ثوابت أخرى لا تعمل مع أوراكل.

Lockedits: ثابت اختياري يقوم بتحديد إقفال مجموعة السجلات. يمكنك اختيار أحد الثوابت المحددة في الجدول ١٦-٦.

الجدول ١٦-٤: ثوابت لنوع المعامل OpenRecordset في مساحة عمل ODBCDirect.

ثابت	الوصف
DbOpenDynamic	يقوم هذا الثابت بفتح وحدة Recordset ديناميكية، التي تشبه خلية ديناميكية لـ ODBC.
DbOpenDynaset	يقوم هذا الثابت بفتح وحدة مجموعة ديناميكية لـ Recordset التي تشبه خلية مجموعة أساسية لـ ODBC.
DbOpenSnapshot	يقوم هذا الثابت بفتح وحدة لقطة لـ Recordset، التي تشبه خلية ثابتة لـ ODBC.
DbOpenForwardOnly	يقوم هذا الثابت بفتح Recordset للأمام فقط.

ملاحظة لم استطع العثور على أي فرق بين خيارات dbOpenDynamic وdbOpenDynaset لنوع خلية Recordset. فكلا النوعان يكتشفا التحديث والحذف الذي تم بواسطة مستخدمين آخرين، وكلا النوعين لا يستطيعان اكتشاف الإدراج. فقد تستخدم dynasets (المجموعات الديناميكية).

الجدول ١٦-٥: الثوابت الممكنة لـ OpenRecordset Options Argument في ODBCDirect Workspace

الثابت	الوصف
DbRunAsync	يقوم هذا الثابت بتشغيل استعلام غير متزامن.
DbExecDirect	يقوم هذا الثابت بتشغيل استعلام بواسطة تخطي SQLPrepare واستدعاء SQLExecDirect مباشرة. وهذا مصمم لخادم SQL وليس له تأثير عند استخدامك لقاعدة بيانات لأوراكل.

ملاحظة SQLPassThrough غير متاحة (أو مطلوبة) في ODBCDirect

الجدول ١٦-٦: الثوابت الممكنة لـ OpenRecordset Lockedits Argument في مساحة عمل لـ ODBCDirect.

الثابت	الوصف
DbReadOnly	يمنع هذا الثابت المستخدمين من إجراء تغييرات في مجموعة السجلات. ورغم أن أوراكل يتحكم في إمكانية تغييرك للبيانات، فإن هذا الخيار يجعل البيانات للقراءة فقط. وهذا الثابت هو الافتراضي لـ ODBCDirect Workspaces.
DbOptimistic	يستخدم هذا الثابت إقفال متفائل، الذي يدل على الصفحة (الصف في حالة أوراكل (التي تحتوي على السجل غير مقفلة حتى يتم تنفيذ أسلوب Update. وهذا هو الافتراض لمصادر ODBCDirect مثل أوراكل.
DbPessimistic	يستخدم هذا الثابت إقفال متشائم، الذي يدل على الصفحة التي تحتوي على السجل مقفلة عند استدعائك أسلوب Edit. لا يعمل القفل المتشائم في مقابل قاعدة بيانات لأوراكل.
DbOptimisticValue	يستخدم هذا الثابت تزامن متفائل استنادا إلى قيم الصفوف
DbOptimisticBatch	يسمح هذا الثابت بتحديث متفائل للمجموعة

وبعد وضع كل هذه الخيارات مع بعضها مرة أخرى، سنحصل على أفضل ثلاث طرق لفتح Recordset: خلايا يمكن تحديثها وللقراءة فقط وللأمام فقط. وتوضح التعليمات البرمجية ١٦-٤ إجراء يتضمن الأنواع الثلاثة.

تعليمات برمجة ١٦-٤

```
Private rsUpdatable As Recordset
Private rsReadOnly As Recordset
Private rsFirehose As Recordset
Private rsMovies As Recordset
```

```
Private Sub CreateRecordsets()
   Dim strSQL As String

   strSQL = "SELECT movie_name, length, rating, year_of_release, " &
_
         " movie_id FROM video_store.movies ORDER BY movie_id"
   Set rsUpdatable = _
      cn.OpenRecordset(strSQL, dbOpenDynaset, 0, dbOptimisticValue)

   Set rsReadOnly = _
      cn.OpenRecordset(strSQL, dbOpenSnapshot, 0, dbReadOnly)

   Set rsFirehose = _
      cn.OpenRecordset(strSQL, dbOpenForwardOnly, 0, dbReadOnly)

'  Select one recordset to work with
   Set rsMovies = rsUpdatable

End Sub
```

ملاحظة يشير توثيق DAO إلى أنه يمكنك تحديد سلسلة استعلام بأكثر من جملة SELECT. وعلى الرغم إمكانية ذلك مع خادم SQL، فإنه لا يمكنك إجراء هذا مع أوراكل.

وعند العمل مع ODBCDirect Recordsets، يكون لديك نفس مجموعة الأساليب التي استخدمتها مع DAO/JET. ولكن يوجد استثناءان:

- ODBCDirect لا يمكنه استخدام أساليب Find، حتى مع خلية قابلة للتمرير. ويرجع هذا إلى أن RDO لا تدعم Find.
- لا يمكنك استخدام أسلوب FillCache. ويرجع هذا إلى أن محرك RDO خلف ODBCDirect أكثر كفاءة من DAO وله CacheSize (حجم ذاكرة وسيطة) معقول للبدء. (ويكون الافتراض ١٠٠ صف). يمكنك فقط تعديل خاصية CacheSize إذا استخدمت QueryDefs.

استخدام (تحديدات الاستعلامات QueryDefs

مع أن تكوين Recordsets بالأساليب الموضحة في المقطع السابق تعمل على نحو لائق، فقد يصعب ترميز وتبطئ عندما تريد استخدام استعلامات تختلف بقيم قليلة في كل مرة. ولتحسين هذا الوضع، يمكننا استخدام QueryDefs. وQueryDefs هي المعادل المباشر لوحدات rdoQuery التي رأيناها في الفصل ١٥ "وحدات البيانات البعيدة". كما تسمح لك بتحديد و"إعداد" استعلام من نوع SELECT مع معاملات بالإضافة إلى، كما سنرى في المقطع "استدعاء الإجراءات المخزنة، إعطاء قوة معاملات المخرجات التي تأتي من RDO.

ولتكوين وحدة QueryDef، قم باستخدام أسلوب CreateQueryDef وصياغته على النحو التالي:

Set *querydef* = *object*.CreateQueryDef *(name, SQL_string)*

والمعاملات هي

name: اسم اختياري لوحدة QueryDef. وكما مع rdoQueries، توجد كمية محددة لتوفير اسم.

SQL_string: نص استعلام SELECT "أو تحديد إجراء مخزن".

سلاسل الاستعلام التي تحددها هي نفسها مثل التي رأيناها مع rdoQueries. فأنت تقوم بتحديد أي معاملات في السلسلة مع متغيرات تعويض ODBC ("؟"). وبالتأكيد ليس عليك استخدام المعاملات إذا لم ترد ذلك، فسيعمل الاستعلام جيدا بدونها، ولكن حينئذ ستفقد قوة QueryDef. وتم توضيح مثال لمعامل QueryDef في التعليمات البرمجية ١٦-٥. وهي مماثلة لتعليمات البرمجة المستخدمة لـ RDO. ولاحظ أنه عند فتح Recordset مقابل QueryDef، فأنت تستخدم نفس المعاملات كأسلوب OpenRecordset لوحدة Connection ما عدا سلسلة الاستعلام.

تعليمات برمجة ١٦-٥

```
Private qy1 As QueryDef
Private Sub CreateQuery1()
    Dim rs As Recordset
    Dim strSQL As String
    strSQL = "SELECT * FROM video_store.movies WHERE "
    strSQL = strSQL & " year_of_release > ? "
```

```
    strSQL = strSQL & " AND rating = ? ORDER BY movie_name"
    Set qy1 = cn.CreateQueryDef("", strSQL)
    qy1.Parameters(0) = 1995
    qy1.Parameters(1) = "PG-13"
    Set rs = qy1.OpenRecordset(dbOpenForwardOnly, 0, dbReadOnly)
   While Not rs.EOF
      MsgBox rs(1)
      rs.MoveNext
    Wend
'   Close the Recordset
    Set rs = Nothing
End Sub
```

تلميح ... لتشغيل استعلام مرة أخرى بعد تغيير قيم المعامل، استخدم أسلوب Requery.

خصائص QueryDef

لوحدة QueryDef خصائص متعددة تسمح لك بالعثور على معلومات معينة عن الوحدة وإعداد زوج من معاملات المؤالفة. وكما ترى في الجدول ١٦-٥، العديد من المعاملات غير متاحة لمستخدمي أوراكل. والأمر المربك هو أن ملفات Visual Basic Help مكتوبة من وجهة نظر SQL Server، ولن تحصل على رسائل خطأ من إعدادك لها. لا تتوقع أن يكون لها تأثير.

والخصائص الأساسية التي يجب عليك ملاحظتها هي CacheSize وذلك للعمل على ملائمة عدد السجلات التي تكون على جانب الخط في اتجاه واحد (خاصة في حالة استخدام مجموعات سجلات كبيرة)، فترة ODBCTimeout التي تعمل على ضمان عدم استمرار الاستعلامات وقتا طويلا، وكذلك StillExecuting والتي تعتبر الطريقة الوحيدة للتحقق من تكوين مجموعة السجلات إذا كنت تقوم باستخدام عملية غير متزامنة.

جدول ١٦-٧: خصائص تحديد الاستعلام QueryDef.

الخاصية	الوصف
Connect	تدل هذه الخاصية على سلسلة الوصلة لـ QueryDef.

جدول ١٦-٧: خصائص تحديد الاستعلام QueryDef.

الخاصية	الوصف
CacheSize	تدل هذه الخاصية على عدد السجلات التي تم تخزينها في الذاكـرة الوسيطة محليا. قم باستخدام هذه الخاصية لموالفة السجلات.
MaxRecords	تدل هذه الخاصية على العدد الأقصـى للسـجلات التـي سـيتم إرجاعها.
Name	تقوم هذه الخاصية بتعيين الاسم الخاص بـ QueryDef في حالـة تحديد اسم له.
ODBCTimeout	تدل هذه الخاصية على الوقت المسموح بالثواني الخاص بالاستعلام وذلك قبل انتهاء الوقت.
Prepare	تقوم هذه الخاصية بالتحكم في ما إذا كان الإجراء المؤقت الذي تـم تخزينه يجب إعداده أم لا (وهذا ليس له تأثير على أوراكل).
RecordsAffected	تدل هذه الخاصية على عدد الصفوف الموجودة في قاعدة البيانـات والتي تأثرت بطريقة التنفيذ الأخيرة. (لا تطبق على أوراكل).
SQL	تدل هذه الخاصية على سلسلة مصدر SQL.
StillExecuting	تدل هذه الخاصية إذا ما كان ما زال يجري تكوين استعلام غـير متزامن.
Type	تقوم هذه الخاصية بتحديد نوع QueryDef مثل جملـة SELECT أو استدعاء إجراء مخزن أو جملة DML.
Updatable	تشير هذه الخاصية إلى ما إذا كان يمكن تحديث Recordset.

استدعاء الإجراءات المخزنة

وبالرغم من استطاعة ODBCDirect تنفيذ أوامر SQL واستدعاءات الإجراءات المخزنة بنفـس طريقة DAO/JET، فإن القوة الحقيقية تكمن في تنفيذ الإجراءات المخزنة مع معـاملات بنفـس طريقة المنتهجة بواسطة RDO. وبطبيعة الحال، تعاني ODBCDirect من نفس مزايا وعيـوب

لأن RDO هو الطريق المؤدي إلى ODBC API. توضح التعليمات البرمجية ١٦-٦ كيفية إعداد QueryDef لتنفيذ إجراء مخزن مع معامل مخرجات. وهي مماثلة لتعليمات البرمجة RDO الذي رأيناه في التعليمات البرمجية ١٥-١٤. في الحقيقة، يمكننا استخدام إجراء فجوال بيسك MovieInsert الذي رأيناه في التعليمات البرمجية ١٥-١٥ لتنفيذ نسخة ODBCDirect هذه.

تعليمات برمجة ١٦-٦

```
Private qyMovieInsert As QueryDef

Private Sub CreateQuery2()
'  Set up any queries required in this module
   Dim strSQL As String
   On Error GoTo ErrorHandler

'  Set up the query to call the insert stored procedure.
   strSQL = "{call video_store.movies_insert3 (?, ?, ?, ?, ?) }"
   Set qyMovieInsert = cn.CreateQueryDef("", strSQL)
   Exit Sub
ErrorHandler:
   If Err = 3146 Then
      MsgBox "Error creating query : " & Errors(0).Description
   Else
      MsgBox "Error creating query : " & Err.Description
   End If

End Sub
```

العمليات غير المتزامنة

وآخر خاصية لنلقي عليها النظر بسرعة هي استخدام العمليات غير المتزامنة. وستتذكر من الفصل ١٥، وحدات البيانات البعيدة، أنه يمكن تشغيل بعض العمليات بطريقة غير متزامنة؛ بمعنى أنك تقوم بإعداد تشغيلها ثم تتابع العمل مع معالجات أخرى. والعملية غير المتزامنة هي

خاصية تم توفيرها بواسطة RDO، ولكن قام فريق التصميم بتوفير خاصية StillExecuting المرتبطة بالعمليات غير المتزامنة. ولسوء الحظ، فإن أحداث RDO 2.0 غير متاحة في ODBCDirect.

يمكنك استخدام العمليات غير المتزامنة باستخدام أساليب ODBC التالية:

- OpenConnection.
- OpenRecordset.
- Requery.
- MoveLast.
- Execute.

في كل حالة من تلك الحالات، يمكنك تحديد ثابت dbRunAsync لمعامل الخيارات. وتعطي التعليمات البرمجية ١٦-٧ مثال قصير عن تكوين Recordset بطريقة غير متزامنة. في هذه الحالة، نستخدم الطريقة الأرخص لاكتشاف إتمام العملية. وإذا أعدت الاطلاع على الفصل ١٥ في المقطع "العملية غير المتزامنة،" ستجد تفاصيل عن استخدام لحدث عداد الوقت لاكتشاف أن العملية قد تمت، وهذا هو الأسلوب المفضل.

تعليمات برمجة ١٦-٧

```
Private Sub CreateAsynchRs()
   Dim strSQL As String

   strSQL = "SELECT movie_name, length, rating, year_of_release, " & 
_
         " movie_id FROM video_store.movies ORDER BY movie_id"

   Set rsMovies = _
      cn.OpenRecordset(strSQL, dbOpenDynaset, dbRunAsync, _
         dbOptimistic)
   If rsMovies.StillExecuting Then
      DoEvents
   End If

End Sub
```

سلبيات ODBCDirect

على الرغم من أن ODBCDirect تعد تحسين كبير على DAO/JET، فإنها لم تحرز تقدم أكثر من هذا وربما ستختفي تدريجيا حينما تصبح ADO متوفرة لكل بيئات التطوير. وعيوب ODBCDirect تشبه عيوب RDO ولكن بشكل أكثر توسعا.

- لا يمكنك استرجاع متغيرات خلايا PL/SQL من الإجراءات المخزنة.
- ومثل DAO و RDO ، تعتبر ODBCDirect تقنية قديمة أنتجتها مايكروسوفت وسيتم تحديثها بأقل تصليح للأخطاء.
- تعتمد ODBCDirect على مشغلات ODBC والمشاكل غير المتوافقة التي تقوم دائما بتكوينها.
- لا تدعم ODBCDirect معاملات ربط أوراكل.
- لا تعتبر ODBCDirect في سرعة RDO و لا ADO.
- لا تكشف ODBCDirect أحداث RDO، ولهذا عليك استخدام الاقتراع لفحص إتمام الأحداث غير المتزامنة.
- لا يمكنك تكوين وحدات منفصلة باستخدام ODBCDirect.

خلاصة

في هذا الفصل، قمنا بتغطية التحسينات التي تدخل على ODBCDirect عبر DAO. ومن بينها وحدة Connection لتحكم أكثر في الوصلات مع قاعدة بيانات أوراكل، وخلايا مصفوفة أكثر عند تكوين Recordsets. واستخدام QueryDefs للاستعلامات مع المعاملات، والإجراءات المخزنة مع معاملات المخرجات. وأخيرا، تعرضنا قدرات ODBCDirect غير المتزامنة.

١٧

وحدات بيانات
ActiveX

تقوم كافة تقنيات الوصول للبيانات التي تناولناها في الكتاب على ODBC. كما رأينا في الفصل ١١ "تقنية جانب المزود"، تم تكوين ODBC لمعالجة مشكلة التوسط لقواعد البيانات العلائقية مثل خادم SQL وأوراكل بشكل عام. هناك حاجة لسحب البيانات من أكثر من مجرد قواعد بيانات علائقية قد تحتاج لبيانات من أنظمة flat-file أو جداول إلكترونية أو بريد إلكتروني أو نظام دفق أعمال أو ربما نظام جمع بيانات ذو سرعة عالية في معمل فيزياء. بالرغم من أن إجابة أوراكل لهذا هي سحب كل شيء إلى قاعدة بيانات عامة، فقد قرر مايكروسوفت أنه من الأفضل توفير إمكانية الوصول إلى كل مصدر من مصادر البيانات الأولية بأسلوب قوي وسريع وشامل أدى ذلك إلى تكوين OLE-DB ومعها وحدات بيانات ActiveX (ADO) التي تجعل من الممكن لمطوري فيجوال بيسك الاستخدام الأمثل لمحرك OLE-DB.

ADO هي واجهة مكونة من ٣٢ بت فقط، تقوم على الاستخدام الآلي OLE وCOM، وبذلك يمكن أن تقوم بنقل البيانات إلى أي بيئة تدعم كلاهما. في الحقيقة، لا يوجد بدائل لاستخدام بيئة مايكروسوفت ويندوز لأنها البيئة الوحيدة التي تدعم فيجوال بيسك.

ADO هي محل جهود مايكروسوفت التطوير للسنوات القليلة القادمة حيث ستظهر إمكانيات جديدة ومثيرة. كما أنها أصبحت تعتبر تقنية للوصول لويب. حقاً، تركت ADO أثرها كتقنية للوصول إلى البيانات الخاصة بـ Active Server Pages (ASP)، [صفحات الخادم النشط]، بيئة scripting تقدم طريقة قوية وفعالة لإجابة طلبات ويب للحصول عن معلومات نشطة وحل للمشكلة أفضل من كل CGI scripting المنتشرة بإصدار فيجوال بيسك ٦، أصبح ADO 2.0 أكثر انتشاراً ليس فقط لبرمجة الإنترنت ولكن أيضاً في التطوير متعدد الطبقات وتطوير الخادم المزود الأكثر تقليدية.

في هذا الفصل، سنتناول موضوعات ADO التالية:

- تدرج وحدات ADO.
- الاتصال بأوراكل.
- العمل مع مجموعات السجلات.
- مجموعات السجلات المفصولة.
- استخدام وحدة الأمر.
- تنفيذ أوامر SQL.
- استدعاء الإجراءات المخزنة.
- العملية غير المتزامنة.
- استرجاع جداول PL/SQL.
- معالجة الأخطاء.

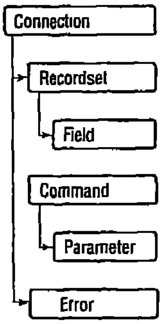
- استخدام بيئة البيانات.
- العوائق التي تقابل ADO.

نموذج وحدة ADO

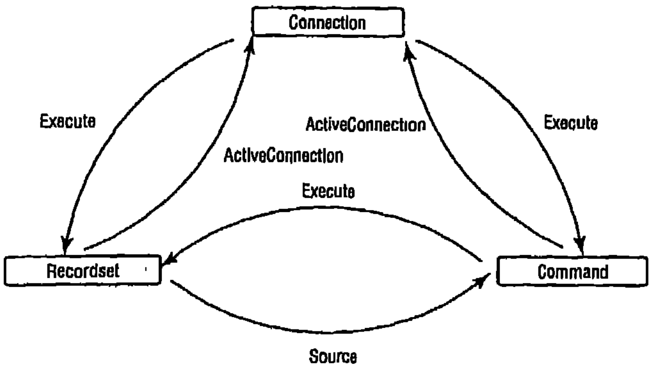
ADO هي انتقال من مجموعة ADO التي تتبعها وحدات أوراكل لـ OLE. كما رأينا مع RDO، هناك مجازفة حيث كلما أصبحت مجموعة الخصائص أكثر قوة كلما أصبح نموذج الوحدة أكثر تعقيدا. في ADO، كلما أصبحت مجموعة الخصائص أكثر قوة كلما أصبح نموذج الوحدة أكثر تعقيداً في ADO، كان الهدف هو إخراج نظام يعتمد على الوحدة ويكون أكثر بساطة ولا يحجب حاجة مطوري الخصائص عند التعامل مع أنظمة مثل أوراكل. النتيجة هي التدرج الذي تراه في الشكل ١٧-١. يختفي التعقيد في وحدات Property "الخواص".

الشكل ١٧-١

تدرج وحدة ADO



يمثل الشكل ١٧-١ التدرج الفعلي لـ ADO، ولكن هناك طريقة بديلة لرؤيـة. في كافة التقنيات الأخرى للوصول إلى البيانات، يجب أن تبدأ بتكوين اتصال بقاعدة البيانات، ثم الوصول إلى قاعدة البيانات التي تريدها وبناء مجموعة سجلات من نوع معين. تسمح لك ADO نظرة على هذه الإمكانية ولكن يجب الآن نلاحظ أن هذه الخاصية تؤدي إلى عرض مختلف لنموذج الوحدة، يعتمد هذا العرض أكثر على الاستخدام العملي للوحدات. هناك طرق عديدة معقدة لتوضيح هذه العملية، ولكن الشكل ١٧-٢ يقوم بتوضيحها ببساطة، مظهراً أن الوحدات الثلاثة الرئيسية يمكن اعتبارها متكافئة بعضها البعض.



الشكل ١٧-٢
النموذج العملي لـ ADO.

يعتبر إحدى مميزات ADO وجود سبعة وحدات فقط هامة في النظام بأكمله. يتم التعريف بهذه الوحدات بعد ذلك.

Connection "الاتصال"

توجد إحدى Connection "الاتصال" أعلى تدرج ADO. بالرغم من أنك تستطيع تكوين وحدات أخرى مستقلة عن وحدة Connection وعن بعضها البعض، لا يمكنك حقيقة عمل أي شيء بدون وحدة Connection، فهي النقطة الوحيدة للوصول إلى مصدر البيانات. مع ADO، يتم الاتصال من خلال مزود OLE-DB (المساوي لبرنامج التشغيل ODBC وDriver Manager (مدير برنامج التشغيل)، ويستطيع هذا المزود الاتصال مباشرة بواجهة OCI الخاصة بأوراكل (تسمى مزود محلي) أو من خلال ODBC. تعتمد إمكانات الاتصال على المزود الذي يتم استخدامه. غالبا تقترن وحدة Connection بصورة مباشرة بوجود Recordset (مجموعات السجلات).

Recordset "مجموعة السجلات"

تعتبر وحدة Recordset "مجموعة السجلات" ثاني أكثر الوحدات أهمية في النموذج، وغالبا ما تكون مركز الإجراء بالكامل الذي يحدث في تفاعلك مع قاعدة البيانات. تتلقى مجموعة السجلات مجموعات النتائج الواردة من الاستعلامات إلى قاعدة بيانات أوراكل وتسمح لك بمعالجتها. مع مزود ODBC، سيمكنك الوصول إلى Recordset التمريرية والقابلة للتحديث. يمكنك تكوين Recordset منفصلة مستقلة عن Connection لكي يتم الاتصال الفعلي بقاعدة البيانات.

Command "الأمر"

وحدة Command "الأمر" هي الوحدة التي تسمح لـــك بالاحتفـــاظ بالاســـتعلامات ومجموعـــة "المعاملات" Recordset وتخزينها ولتنفيذ الإجراءات المخزنة داخل معاملات إدخال وإخـــراج. تعتبر مشابهة لــ rdoQuery في RDO. يمكنك تنفيذ أمر على Recordset دون تكوين وحـــدة Command بالرغم من أنه سيتم تكوين واحدة ضمنيا تأتي وحدة Command بكل الخصـــائص الجديدة المتاحة من ADO ومزود OLE-DB اللذين تقوم باستخدامهما.

Parameter "المعاملات"

يتضمن نموذج ADO مجموعة متنوعة من المعاملات كم هو الحال مـــع RDO بالنســبة لكــل Parameter. لديك خيار تكوين مجموعات Parameter "المعاملات" المستقلة، وتعتبر هذه تقنية مفضلة للحصول على برمجة فعالة.

Field "الحقل"

تمثل وحدة Field "الحقل" عمود واحد في represents. عند التعامل مع أوراكل، سيتم تخطيــط كل Field إلى عمود واحد في صف في مجموعة نتائج الاستعلام. تحتوي وحـــدة Field علـــى معلومات أكثر من RDO أو DAO بالإضافة إلى معلومات كثيرة كما سترى من OO40. يوضع كل Field داخل كل Recordset.

Error "الخطأ"

تشبه وحدة Error ما رأيناه مع DAO وRDO. توضع في مجموعة Errors وتمثل خطأ وصـــول البيانات الأكثر حداثة، تمثل وفي هذه الحالة الرسالة الواردة من خادم أوراكل أو بعض الأمـــاكن الوسيطية على مدار الطريق. تستطيع مجموعة Errors احتواء أكثر من خطأ واحـــد وارد مـــن نفس الاستدعاء إلى أوراكل.

Property "الخواص"

كما ذكرنا في المقدمة، تم وضع معظم قوة ومرونة ADO في وحـــدات Property "الخـــواص". يسمح ذلك بتحديد قيم مجموعة متنوعة من سمات الواجهة. بالرغم من أن الخواص هي تلك التي تم بناءها في ADO، يوجد أيضا إمكانية وجود خواص أخرى خاصة بمـــزود OLE-DB الـــذي تستخدمه. سيؤدي ذلك إلى زيادة الفعالية لمصدر بيانات معين مع تغيير في قدرات المـــزود، لا تحتاج الوحدة إلى تغيير كامل.

الاتصال بأوراكل

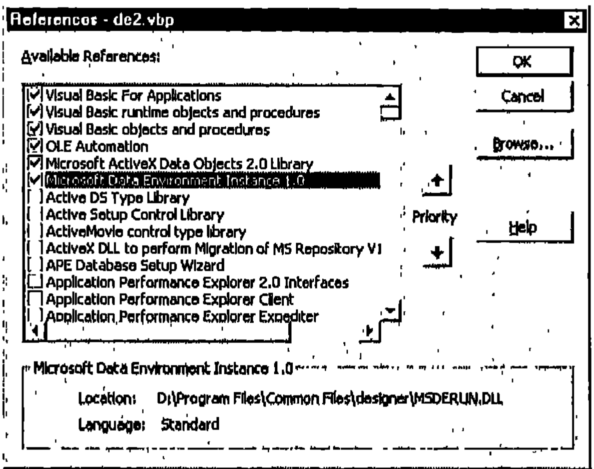
كلما نتوغل في إمكانيات ADO، سنجد أنك تستطيع تكوين أي وحدة من الوحدات الرئيسية الثلاثة "Connection و Recordset و command" مستقلين عن الوحدات الأخرى. يمكـــن أن يـــؤدي ذلك إلى تبسيط تعليمات البرمجة إلى مدى معين لأنك تستطيع تكوينه في سطر واحـــد لتعليمـــات البرمجة التي تحتاج لأسطر عديدة في أي تقنية أخرى للوصول إلى البيانات. على أي حال، هناك شيئاً يجب أخذه في الاعتبار وهو أن أحياناً ما تكون البساطة والفعالية أهدافاً متعارضـــة تمامـــاً. إليك نصيحة جيدة وهي الاعتياد على كافة التقنيات الخاصة بإجراء اتصال، مثل تكوين اتصـــالاً ظاهريا، بالإضافة إلى مميزاتها، ثم تستطيع بعد ذلك اللجوء إلى التقنيات الأكثر بساطة.

للاستفادة من أي خاصية من خصائص Ado، يجب أولا إعداد مرجع في التطبيـــق لمكتـــبة ADO في المشروع الحالي. يمكننا تحقيق ذلك من خلال الخطوات الآتية:

١- اختر Project ⇦ References لفتح مربع الحوار References كما هو موضح فـــي الشكل ١٧-٣. سترى قائمة بكافة مكتبات الوحدات التي يمكنك تزويدها بمرجع التطبيق.

٢- حدد الخيار المسمى Microsoft ActiveX Data Objects 2.0.

٣- انقر OK.



الشكل ١٧-٣ تحديد خيار مكتبـــة ADO في مربـــع الحـــــوار Reference.

أول ما ستقوم بعمله هو تكوين وحدة Connection ظاهريا، بمعنى آخر ســـنكتب تعليمـــات البرمجة لتكوين وحدة Connection بدلا من تكوينها ضمنيا من خلال وحـــدات Recordset أو Command. بهذه الطريقة، يمكننا تكوين وحدة Connection واحدة لك قاعدة بيانات نســـتطيع الوصول إليها، كما يمكننا استخدام هذا الاتصال للتفاعل الكامل مع قاعدة البيانات. كمـــا ســـنرى

لاحقا، إذا قمنا بتكوين اتصال ضمنيا، فسنحصل إذن على اتصال جديد ومختلف بقاعدة البيانات في كل مرة.

إذا نظرت إلى الجدول ١٧-١، سترى الأساليب المختلفة التي يمكنك استخدامها مع وحدة Connection.

الجدول ١٧-١: أساليب وحدة Connection

الأسلوب	البيان
BeginTrans, CommitTrans, RollbackTrans	تقوم هذه الأساليب بمراقبة الحركات من ADO. لاحظ أن التثبيت الآلي (autocommit) هو الحدث الافتراضي لوحدة Command.
Cancel	يقوم الأسلوب بإلغاء تنفيذ أسلوب Execute أو Open غير المتزامن.
Close	يقوم هذا الأسلوب بإغلاق الاتصال وأي Recordset مقترنة.
Execute	يقوم هذا الأسلوب بتنفيذ استعلام أوامر SQL المحدد.
Open	يقوم هذا الأسلوب بفتح الاتصال بقاعدة البيانات المحددة.
Open Schema	يعيد هذا الأسلوب Recordset توفر معلومات تفصيلية عن قاعدة البيانات مثل الجداول والأعمدة.

في هذا القسم، سنطلع على أسلوب Open الذي يسمح لنا بإعداد اتصالا جديدا بقاعدة بيانات أوراكل. يمكننا تحديد بيان الاتصال الذي تريد عمله عن طريق الخصائص أو سلسلة وصل أو كلاهما. دعنا نأخذ هذه الخطوة الواحدة في كل مرة.

في تعليمات البرمجة ADO، نقوم بشكل نموذجي بتكوين وحدة مستقلة وتعيين المعاملات الخاصة بها ثم فتحها. نقوم بتكوين وحدة Connection عن طريق أحد الأسلوبين التاليين:

```
Dim cn As New ADODB.Connection
```

أو

```
Private cn As ADODB.Connection
```

```
Set cn = New ADODB.Connection
```

ملاحظة عندما تستخدم ADO، تبدو التعريفات أكثر تعقيدا شيئا ما عندما يتم الإشارة إلى أنواع المتغيرات خلال مكتبة ADODB-TYPE

بالرغم من سطر الرموز الإضافي، أفضل الشكل الثاني من التعريف، وهو التكوين الظاهري بعكس التكوين الضمني. من بين المميزات الأخرى، يتيح لك الأسلوب الثاني فرصة استخدام الكلمة الأساسية With Events، كما أنه يسمح لك بالمراقبة الظاهرية على تكوين الوحدة.

لدينا هنا اتصالا مستقلا، ولكن لا يمكننا استخدامه للوصول إلى البيانات حيث أنه لم يتم فتحه بعد. نقوم بتعيين خصائص الاتصال قبل فتحه. يتم توضيح الخصائص المتاحة في الجداول ١٧-٢. يؤدي استخدام القيم الموجودة في هذا الجدول إلى توفر مراقبة على الاتصال أكثر من المراقبة عند تكوين اتصال ضمني بوحدة Recordest. كما ترى، تعتبر بعض الخصائص أكثر أهمية من خصائص أخرى، خاصة بالنسبة لمستخدمي أوراكل.

الجدول ١٧-٢: خصائص الاتصال.

الخاصية	البيان
Attributes	تحدد هذه الخاصية ما إذا قامت CommitTrans أو Rollback ببدء حركة جديدة (انظر ملف التعليمات الخاص بفيجوال بيسك للإطلاع على الثوابت).
CommandTimeout	توضح هذه الخاصية وقت انتظار حتى ينتهي (الوقت الافتراضي هو ٣٠ ثانية).
ConnectionString	توضح هذه الخاصية المعاملات الخاصة ADO. إذا قمت بتوفير معاملات أخرى في هذه السلسلة، يتم إرسالها إلى المزود (على سبيل المثال، إلى برنامج تشغيل ODBC).
ConnectionTimeout	توضح هذه الخاصية وقت انتظار اتصال حتى ينتهي (الوقت الافتراضي هو ١٥ ثانية).

الجدول ١٧-٢: خصائص الاتصال.

الخاصية	البيان
CursorLocation	تستخدم هذه الخاصية adUseClient لتحديد مؤشرات تعتمد على المزود وadUseServer للمؤشرات الموجودة علـــى الخـــادم أو المزود بالنسبة لأوراكل، يجب استخدام adUseClient.
DefaultDatabase	توضح هذه الخاصية قاعدة البيانات الافتراضيـــة عندمـــا يمكـــن للمزود استخدام أكثر من قاعدة بيانات لكل اتصال.
IsolationLevel	توضح هذه الخاصية عزل الحركات الخاصة بالاتصال
Mode	تقوم هذه الخاصية بتعيين أو إعادة تصريحات بالوصول، تجـــاهل هذه الخاصية بالنسبة لقواعد بيانات أوراكل
Provider	توضح هذه الخاصية اسم مزود OLE-DB. (يمكن أيضاً تعييـــن ذلك في خاصية ConnectionString
State	تشير هذه الخاصية إلى أن الاتصال مفتوحاً (adStateClose) أو مغلقاً (adStateClosed)
Version	توضح هذه الخاصية رقم إصدار ADO

في حالة استخدام خصائص وحدة Connection الأكثر أهمية، يمكننا البدء في بناء إجـــراء ConnectToDatabase (انظر التعليمات البرمجية رقم ١٧-١). كما تـــرى فـــي التعليمـــات البرمجية، نقوم بتكوين وحدة Connection جديدة مستقلة "CN" ذات قيمة انتهاء الوقـت (Time-out value) مقدارها ثلاثة ثواني ومؤشر يعتمد على المزود (وهو الوحيد المتاح لأوراكل)، ثـــم نقم بتحديد مزود OLE-DB الأصلي لأوراكل (MASAORA).

تعليمات برمجة ١٧-١

```
Private cn As ADODB.Connection
Private Sub ConnectToDatabase()
'  Connect to the Oracle database
```

```
    Set cn = New ADODB.Connection
    With cn
       .ConnectionTimeout = 3
       .CursorLocation = adUseClient
       .Provider = "MSDAORA"
    End With
 End
```

لدينا الآن Connection مستقلة، نحن مستعدون إذن لفتح هذا الاتصال. إليك صيغة أسلوب Open (الفتح):

connection.Open ConnectionString, UserID, Password, OpenOptions

والمعاملات الاختيارية التالية:

ConnectionString: تحتوي هذه السلسلة على معلومات عن الاتصال ستقوم أي معلومات في هذا المعامل بإلغاء إعدادات خاصية ConnectionString.

UserID: تحتوي هذه السلسلة على اسم مستخدم أوراكل. إذا تم توفيره، سيقوم هذا المعامل بإلغاء اسم المستخدم الموجود في معامل ConnectionString.

Password: كلمة السر هذه خاصة باسم المستخدم الذي تم توفيره. إذا تم توفيره، سيقوم هذا المعامل بإلغاء كلمة السر الموجودة في ConnectionString.

OpenOptions: يجب تعيين هذا المعامل إلى adConnectAsync لكي يتم فتح الاتصال بطريقة غير متزامنة.

هناك العديد من الطرق المختلفة لتعبئة هذه المعلومات مثلما يستطيع كل معامل أو سطر تالي تجاوز قيم سابقة. حدد أسلوب معين واحد والتزم به كمعيار موقع. إذا فعلت شيئاً غير ذلك فعندما تحتفظ بتعليمات برمجة لشخص آخر، ارتباك الاتصال. ستطلع على طريقتين لإجراء الاتصال، إحداهما استخدام كل مزود من مزودي OLE-DB المتاحين المزود لـ ODBC والمزود الأصلي لأوراكل.

مزود OLE-DB لـ ODBC

كان ADO قليل الفائدة لأي فرد إذا لم يكن لديك مزود OLE-DB لكي تستخدمه في بيئتك. كانت أبسط طريقة لمايكروسوفت لتوفير مزود OLE-DB إنتاج مزود لـ ODBC. يعتبر مزود -OLE

DB لـ ODBC (MSDASQL) المزود الافتراضي. بالرغم من أن مزود OLE-DB يوفر طبقة أخرى بين تعليمات البرمجة وأوراكل إلا أن يقتصر بعض المميزات. يكشف المزود عن كل وظائف ODBC، وكما سنرى لاحقاً يوفر لنا بصورة مذهلة إمكانية وصول أفضل إلى الخصائص وأحياناً أداء أفضل من RDO.

حيث أن مزود ODBC هو المزود الافتراضي، لا يجب مطلقاً تحديده. إذا اعتبرنا أن OLE-DB ستقوم أيضاً بالتمرير على أي معاملات لا تعرفها إلى أسفل السلسلة (في هذه الحالة، برنامج تشغيل ODBC)، يمكننا استخدام نفس سلسلة الوصل بالتمرير التي قمنا باستخدامها لـ ODBC كما يلي:

```
cn.Open "DSN=movies;uid=video_user;pwd=ummagumma;"
```

وأخيرا، سنجمع ذلك في إجراء ConnectToDatabase (انظر التعليمات البرمجية رقم ١٧-٢). سنقوم باستخدام مزود ODBC في غالبية هذا الفصل بسبب الوظيفة الإضافية الذي نريد الإطلاع عليها، وأيضا لأنه أكثر نضجا وأقل تعرضا للخطأ من المزود الأصلي.

تعليمات برمجة ١٧-٢

```
Private cn As ADODB.Connection
Private Sub ConnectToDatabase()
' Connect to the Oracle database
   Set cn = New ADODB.Connection
   With cn
      .ConnectionTimeout = 3
      .CursorLocation = adUseClient
   End With
   cn.Open "DSN=movies;UID=video_user;PWD=ummagumma;"
End
```

يمكننا أيضا استخدام اتصال DSN-less مع ADO كما فعلنا مع RDO. تقدم التعليمات البرمجية رقم ١٧-٣ مثالا عن كيفية عمل ذلك. تذكر أن مميزات اتصالات DSN-less هي أنه لا يجب تكوين DSN "اسم مصدر البيانات المسجل" على كل جهاز مزود، وأنها أسرع بصورة طفيفة حيث أن تصل إلى السجل لاكتشاف اسم خدمة أوراكل الذي يجب الوصول إليه. تعتبر ADO سريعة في إجراء الاتصالات، ولكن لا مانع من جعلها أكثر سرعة.

تعليمات برمجة ١٧-٣

```
Private cn As ADODB.Connection
Private Sub ConnectToDatabase()
'  Make a DSN-less connection to the Oracle database
   Dim strConnection As String

   Set cn = New ADODB.Connection
   With cn
      .ConnectionTimeout = 3
      .CursorLocation = adUseClient
   End With

   strConnection = "SERVER=movies;" & _
            "DRIVER={Microsoft ODBC for Oracle};" & _
            "UID=video_user;PWD=ummagumma;"
   cn.Open strConnection
End
```

مزود OLE-DB الأصلي لأوراكل

بالرغم من أن مزود OLE-DB لـ ODBC هو المزود الذي ظهر أولاً (وهو المزود الأكثر أهمية لقبول ADO)، تم إصدار فيجوال بيسك ٦ بمزودين OLE-DB أصليين لأوراكل وخادم SQL ضمن الآخرين. أصبح OLE-DB فعلاً مقياساً مقبولاً مثل ODBC، سيحدث مزيد من التقدم في هذا الجانب بالإضافة إلى النتائج (التداعيات) المثيرة. في الوقت الحالي، بالرغم من أن لدينا مزود أصلي يأتي بين مستوى مستهلك بيانات ADO وواجهة الاستدعاء لأوراكل، فإن هذا المزود لديه وظيفة تقارن بمزود ODBC، ولكن في نظام High-END، من المحتمل ألا تكن مستخدماً هذه الوظيفة (مثل المؤشرات القابلة للتمرير والتحديث) على أي حال.

لاستخدام مزود OLE-DB لأوراكل تحتاج إلى تحديد اسم المزود ("MSDAORA") بالإضافة إلى باقي معلومات الاتصال. يوضح السطر التالي مثالاً لذلك.

```
cn.Open "Provider=MSDAORA;Data Source=movies;" & _
      "User Id=video_user;Password=ummagumma;"
```

في هذه السلسلة، لاحظ أن الأزواج لقيم الكلمات الأساسية مشابهة لتلك التي تستخدم في ODBC.

المزود: نقوم بإدراج اسم المزود هنا، بالرغم من إننا نستطيع بسهولة تحديد هذا بواسطة خاصية المزود الخاصة بوحدة Connection.

مصدر البيانات: هذه الكلمة الأساسية اسم خدمة أوراكل (بديل قاعدة البيانات).

معرف المستخدم: تحدد هذه الكلمة الأساسية اسم المستخدم الخاص بالاتصال.

كلمة السر: تحدد هذه الكلمة الأساسية كلمة السر الخاصة بالاتصال.

ملاحظة لاحظ اختلاف تعريف المستخدم "User ID" والكلمات الأساسية لكلمات المرور "Password" عن ODBC التي تستخدم كل من VID وPWD.

توضح التعليقات البرمجية رقم ١٧-٤ مثالاً لإجراء الاتصال باستخدام مزود OLE-DB الأصلي لأوراكل. النقاط الرئيسية التي يجب ملاحظتها هي أننا نقوم بتعريف اسم المزود وأنه يجب أن نضمن أن سلسلة الوصل يتم تعريفها بالكلمات الأساسية المناسبة.

تعليمات برمجة ١٧-٤

```
Private cn As ADODB.Connection
Private Sub ConnectToDatabase()
'  Connect to the Oracle database
   Set cn = New ADODB.Connection
   With cn
      .ConnectionTimeout = 3
      .CursorLocation = adUseClient
      .Provider = "MSDAORA"
   End With
   Cn.Open "Data Source=movies;User
Id=video_user;Password=ummagumma;"
End Sub
```

تحذير لم تثبت الإصدارات الأولى لمزود OLE-DB لأوراكل استقرارها. أنتبه إلى التأكد من ترميز سلسلة الوصل بصورة سليمة، وإلا فقد تلاحظ توقف التطبيق عن العمل.

أحداث الاتصال

كانت إحدى نقاط ضعف ADO 1.5 مقارنة بـ RDO هي أنها الأقل في معالجة الحدث. مع ADO 2.0، التي تم إصدارها مع فيجوال بيسك ٦، يوجد نموذج حدث يناقش، إن لم يفوق RDO 2.0 كما سترى في الجدول ١٧-٣، توفر لك هذه الأحداث سيطرة كبيرة على عملية وسيط البيانات. يتم إطلاق أغلب هذه الأحداث حتى لو لم تكن العملية ناجحة. بالرغم من أن أحداث عديدة ستخطي بأدنى تقدير لمستخدمي أوراكل، فستثبت أحداثاً قليلة أنها مفيدة كما هو الحال مع RDO.

الجدول ١٧-٣: أحداث اتصال ADO.

الحدث	البيان
Begin Trans Complete	إتمام عملية Begin Trans (بدء حركة).
Commit Trans Complete	إتمام عملية Commit Trans (تثبيت حركة).
Connect Complete	إتمام عملية Connection (اتصال)
Disconnect	إتمام عملية Disconnect (قطع اتصال).
Execute Complete	إتمام أسلوب (Execute) أمر في هذا الاتصال.
Info Message	رسالة معلومات تم إضافتها لمجموعة الأخطاء (خادم SQL فقط)
RollbackTrans Complete	إتمام عملية RollbackTrans (إلغاء تعديلات الحركة).
WillConnect	ليتم إطلاق الحدث قبل بدء محاولة اتصال.
WillExecute	يتم إطلاق الحدث قبل تنفيذ استعلام في هذا الاتصال.

لاستخدام أحداث مع ADO، تحتاج إلى استخدام نفس التقنيات التي رأيناها مع RDO. يجب أولاً الإعلان عن وحدة الكلمة الأساسية With Events إذا كنت تريد استخدام الأحداث الخاصة بها. يتم توضيح أمثال لهذا الإعلان في السطر التالي:

```
Private WithEvents cn As ADODB.Connection
```

في حالة استخدام الكلمة الأساسية WithEvents، لا يمكنك استخدام كلمة الأساس New في نفس الإعلان، سيجب استخدام أمر Set لتكوين الوحدة الجديدة كما فعلنا في الأمثلة القليلة السابقة. يعتبر هذا التقييد كافياً لوضع نمط الإعلان خلال التطبيق.

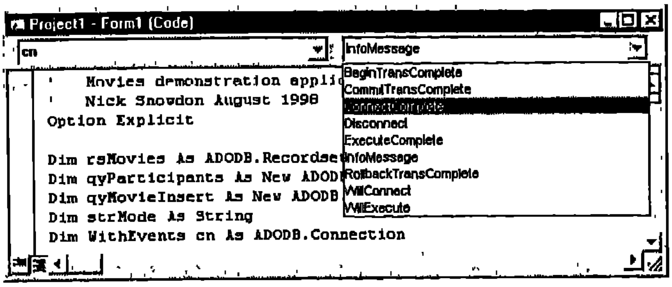
ملاحظة يجب استخدام كلمة الأساس WithEvents للإعلان عن الوحدات على مستوى الوحدة النمطية وليس على المستوى الفرعي.

في التعليقات البرمجية رقم ١٧-٢٥ أن ترى مثالاً أكثر كمالاً لاستخدام الأحداث. تقوم تعليمات البرمجة بتعريف الاتصال cn باستخدام WithEvents. ثم نقوم بتكوين Connection مستقلاً جديداً في معالج الحدث Form_Load وتقوم بفتحه بمجرد الإعلان عن أي وحدة باستخدام WithEvents. يمكنك إذن استخدام مربعات فيجوال بيسك المنسدلة لتكوين الأحداث التي تحتاجها بالصيغة الصحيحة (انظر الشكل ١٧-٤). تعتبر هذه التقنية أسهل من محاولة ترميز الأحداث يدوياً، حيث لا يجب فقط أن يكون اسم الحدث صحيحاً (ويتم فصل الوحدة واسم الحدث بخط وسيط) ولكن يجب أيضاً أن تكون صيغة معاملات الحدث صحيحة.

تعليمات برمجة ١٧-٥

```
Private WithEvents cn As ADODB.Connection
Private Sub Form_Load()
   Set cn = New ADODB.Connection
cn.ConnectionTimeout = 4
   cn.CursorLocation = adUseServer
   cn.Open "DSN=movies;uid=video_user;pwd=ummagumma;", "", "",
_
         adConnectAsync
End Sub
Private Sub cn_ConnectComplete(ByVal pError As ADODB.Error, _
         adStatus As ADODB.EventStatusEnum, _
         ByVal pConnection As ADODB.Connection)
   If adStatus = adStatusOK Then
      MsgBox "Connected OK"
   Else
```

```
    MsgBox "Error connecting " & pError.Description
  End If
End Sub
```



الشكل ١٧-٤ تكوين معالج الحدث.

يعتبر حدث ConnectComplete الذي تراه في التعليمات البرمجية رقم ١٧-٥ تابع لأحداث ADO حيث أنه يقدم معلومات أكثر من أحداث RDO المماثلة. تتضمن المعلومات وحدة Error لتفسير الخطأ لأي فشل في الاتصال ووحدة Status والاتصال الذي وقع الحدث عليه. إن وحدة Status ممتعة حيث أنها توفر لنا عدداً من قيم الحالة الممكنة.

AdStatusOK: الحالة إذا كان الاتصال ناجحاً.

AdStatusErrorsOccurred: الحالة إذا فشل الاتصال.

AdStatusCancel: الحالة إذا تم إلغاء الاتصال في الحدث WillConnect.

AdStatusUnwantedEvent: قم بتعيين هذه الحالة لتجنب أي رسائل أحداث أخرى.

العمل مع Recordsets "مجموعة السجلات"

كما رأينا DAO، نحن نحتاج لتكوين Recordsets لمعالجة مخرجات استعلامات SELECT إلى قاعدة بيانات أوراكل. الآن مع ADO، لدينا وسائل عديدة لإجراء ذلك، وسوف نطلع على كافة هذه الوسائل. على أي حال، إذا قمت بالعمل مع أي أساليب أخرى للوصول إلى البيانات، سيكون الأسلوب الأكثر شيوعاً هو استخدام أسلوب Open الذي يوفر لنا أكبر سيطرة على تكوين Recordsets.

نقوم مرة أخرى باستخدام التقنية التي عادة ما نراها في ADO، وهي تكوين وحدة Recordsets مع كلمة الأساس New كما يلي:

```
Set rs = New ADODB.Recordset
```

لا يحتوي ذلك على أية سجلات. لا يمكننا عمل أي شيء به إلى أن تقوم باستخدام أسلوب Open بالصيغة التالية:

rs.Open Source, ActiveConnection, CursorType, LockType, Options

الأسلوب لديه المعاملات الاختيارية التالية:

Source: هذا المعامل عبارة عن سلسلة SQL SELECT.

ActiveConnection: يمكن أن يكون هذا المعامل إما وحدة **Connection** نشطة أو معامل ConnectString ذات معلومات كافية لإجراء اتصالاً جديداً.

CursorType: إذا كنت تعمل مع مزود بيانات ODBC، يمكننا استخدام مزود مؤشر ADO لتوفر لنا الاختيارين أربعة مؤشرات "انظر الجدول ١٧-٤".

LockType: يوفر المعامل نوع الإقفال الذي نريده. يتم توضيح الاختيارات في الجدول ١٧-٥.

Options: يوفر هذا المعامل الخيارات الممكنة التي يمكنك تضمينها مع أسلوب Open كما هو مفصل في الجدول ١٧-٦.

الجدول ١٧-٤: القيم الممكنة لمعامل CursorType (نوع المؤشر).

الثابت	نوع المؤشر
AdOpenForwardOnly	مؤشر للأمام فقط (الافتراضي)
AdOpenKeyset	مؤشر تتحكم فيه مجموعة من المفاتيح
AdOpenDynamic	مؤشر متحرك
AdOpenStatic	مؤشر ثابت

الجدول ١٧-٥: القيم الممكنة لمعامل Locktype (نوع الإقفال).

الثابت	نوع الإقفال
AdLockPessimistic	تزامن تشاؤمي "لا يتناسب مع أوراكل"
AdLockReadOnly	تزامن تفاؤلي باستخدام قيم صفوف
AdLockOptimistic	مؤشر للقراءة فقط "الافتراضي"

الجدول ١٧-٥: القيم الممكنة لمعامل Locktype (نوع الإقفال).

الثابت	نوع الإقفال
AdLockBatchOptimistic	يؤجل أي تحديثات إلى أن تستخدم طريقة BatchUpdate "تحديث مجموعة"

الجدول ١٧-٦: القيم الممكنة لطريقة Open

الثابت	البيان
AdCmdText	سلسلة Source تحتوي على أمر SQL مثل جملة SELECT.
AdCmdTable	سلسلة Source تحتوي على اسم جدول يجب استرجاعه. تقوم ADO بإنشاء استعلام SQL). لا يفضل استخدام الأمر لأنه يمكن أن يجلب سجلات كثيرة جداً.
AdCmdStoredProc	سلسلة Source تحتوي على اسم إجراء مخزن (لا يتم تطبيق ذلك على أوراكل).
AdCmdUnknown	يدل هذا الثابت على أن نوع الأمر في معامل Source غير معروفاً. يعتبر استخدام هذا الخيار أمراً غير فعالاً.
AdExecuteAsync	يدل هذا الثابت على أن سلسلة Source يتم تكوينها بطريقة غير تزامنيه.
AdFetchAsync	يدل هذا الثابت على أن ado يجب أن تقرأ عدد السجلات في CacheSize أولاً ثم تقوم بملء الباقي بصورة غير تزامنيه.

كما هول الحال مع RDO وDAO، لدينا حقيقة عدداً محدوداً من أنواع مجموعة السجلات المختلفة لكي يتم فتحها، وذلك بناءاً على ما إذا كنا نريد أن تكون مجموعة السجلات قابلة للتحديث أو قابلة للتمرير أو للأمام فقط. يعتبر مؤشر ADO الافتراضي مؤشر للأمام فقط وللقراءة فقط، كما يعتبر هذا المؤشر أرخص "أكثر فعالية" مؤشر حيث أنه لا حاجة لتخزين البيانات بالإضافة إلى أنه متماشياً مع الطريقة التي يريدها أوراكل لاستعادة مجموعة النتائج. إذا

كنت بحاجة إلى خصائص قابلية التحديث الخاصة بمجموعة سجلات، فليس لديك خيـار، إذن، إلا اختيار مزود OLE-DB لـ ODBC، حيث أن المزود الأصلي هو مؤشر للأمام فقـط وللقـراءة فقط. توفر التعليمات البرمجية رقم ١٧-٦ مثالاً لكل نوع من مجموعة السجلات القابلة للتمريـر والقابلة للتحديث، والقابلة للتمرير وللقراءة فقط، وللأمام فقط وللقراءة فقط. في الحالة الأخـيرة، ليس عليك إدخال المعاملات لـ CursorType وlockType حيث سيتم افتراضـها إلـى القيـم المطلوبة.

تعليمات برمجة ١٧-٦

```
Private WithEvents rs1 As ADODB.Recordset
Private WithEvents rs2 As ADODB.Recordset
Private WithEvents rs3 As ADODB.Recordset

Private Sub CreateRecordsets()
'  Create three types of Recordset
'  Scrollable and updatable
   Set rs1 = New ADODB.Recordset
   rs1.Open "SELECT * FROM movies", cn, adOpenKeyset,
adLockOptimistic
'  Scrollable and read-only
   Set rs2 = New ADODB.Recordset
   rs2.Open "SELECT * FROM movies", cn, adOpenStatic,
adLockReadOnly

'  Forward-only and read-only
   Set rs3 = New ADODB.Recordset
   rs3.Open "SELECT * FROM movies", cn, adOpenForwardOnly, _
         adLockReadOnly
End Sub
```

هل يجب أن تقوم مجموعة السجلات بتكوين اتصال

هناك اختلافا عن الكيفية التي تتبعها لتكويـن مجموعـة السـجلات، مـن المحتمـل أن تكون قد سمعت الكثير عن إمكانية تكوين سجلات وعـن تكويـن اتصـال أليـا.

يمكنك إجراء ذلك عن طريق تمرير سلسلة الوصل بدلا من وحدة Connection (وهي cn في الأمثلة). يوضح الرمز التالي مثالا لإجراء ذلك:

```
Dim rs4 As New ADODB.Recordset
Rs4.Open "SELECT * FROM movies", _
  "DSN=movies;uid=video_user;pwd=ummagumma;"
```

بالرغم من أن هذا الرمز يعتبر طريقة نمطية لتكوين مجموعة سجلات (ويوصي بأن AO ليس نظاما تدريجيا)، يجب أن تضع في اعتبارك أن ذلك سيؤدي دائما إلى تكوين وحدة Connection جديدة حتى لو كانت سلسلة الوصل هي نفس سلسلة الوصل المستخدمة سابقا. إذا أين تستخدم هذه التقنية؟ قم بقصر استخدامها على المواقف التي تقوم فيها بتكوين مجموعة سجلات واحدة في التطبيق. قد يكون ذلك استرجاعا، مستندا إلى ويب، لكمية قليلة من البيانات يجب أيضا إذن محاولة الاحتفاظ باتصالا واحدا لكافة الطلبات. إن أحد أسباب الابتعاد عن DAO هو الحصول على سيطرة أفضل على الاتصالات، وليس سيطرة أسوأ. استخدم هذه التقنية.

طرق أخرى لتكوين مجموعات سجلات

يعتبر أسلوب Open على وحدة Recordset هو أفضل طريقة لتكوين مجموعة سجلات، وذلك لأنك تستطيع تضمين ليس فقط نوع المؤشر ولكن نوع الإقفال (التزامن) فقط. على أي حال، لا يمكننا الاستمرار حتى تذكر أسلوبين آخرين يمكنهما إنتاج مجموعة سجلات كمخرج. هذه الأساليب هي:

- Connection.Execute.
- Command.Execute.

ينتج كل أسلوب مجموعة سجلات بمؤشر ثابت للأمام فقط. في أحوال عديدة، قد يكون هذا الموقف هو ما تريده، ولكن هذا الاقتصار هو شيئا يجب أن تضعه في اعتبارك.

استكشاف أساليب وخصائص مجموعة السجلات

بالرغم من قيامك بتكوين مجموعة السجلات بأسلوب Open، لديك نطاق مماثل من الأساليب لكي تستخدمها مع مجموعة السجلات التي رأيناها مع RDO. يفضل هذا القيم العديد من العمليات المختلفة التي ستحتاج إلى تنفيذها، كما يقدم قسم "التجميع كلية" التابع لهذا الفصل، مثالا متكاملا لأغلب هذه العمليات.

ملء مجموعة السجلات

عند فتح مجموعة سجلات، تقوم ADO بملئها "أي ملئها ببيانات فعليـــة" حتـى قيمـة خاصيـة CacheSize "حجم الذاكرة الوسيطة". القيمة الافتراضية لهذه الخاصية هي ١، لذلك يتم قـــراءة سجل واحدة فقط من مجموعة نتائج أوراكل. يتم تخزين محليا في ذاكرة المزود. عندما تنتقل إلى سجل جديد، ستقوم ADO أولا بالتحقيق من وجوده بالذاكرة الوسيطة. في حالــة عـدم وجـوده، ستقوم ADO بملء الذاكرة الوسيطية مرة أخرى.

من الواضح أن الحجم الافتراضي ١ للذاكرة الوسيطة نادرا ما يكون فعالا. يعني كل ســـجل جديد يتم قارئته جولة أخرى على الشبكة، الأمر الذي سيؤدي إلى زيادة المصروفات الإضافية.

بالرغم من هذه القيمة الافتراضية، يجب دائما تحديد قيمة تتناسب مع عدد الســـجلات التـي تتوقع قراءتها. ضع في اعتبارك أن الجزاء هو الحاجة لذاكرة أكبر بجانب المــزود. سـيقتضي عليك الاختيار بواسطة قيم مختلفة لـ CacheSize. توضح التعليمات البرمجيـــة رقـم ١٧-٧ إجراء اختبار قمت باستخدامه لتحديد موقف التأثيرات. في الواقع، تعتمد الفوائـد علـى الشـبكة معدات الجهاز الخاص بك. إذا قمت بتجربة ذلك على شبكة ذات تحميل شديد، ستكون تـــأثيرات CacheSize الكبيرة أكثر إثارة. حتى إذا كانت لديك ســـجلات قليلـــة فقـط، يجـب اسـتخدام CacheSize ذات حجم أكبر من ١ لأن ذلك سيخفض المحمل على الشبكة. عندما تكـون لديـك المئات من السجلات، يجب حتماً ضبط المعامل بالإضافة إلى معرفـــة سـبب احتيــاجك لـــهذه السجلات الكثيرة.

تعليمات برمجة ١٧-٧

```
Private Sub TestCache(iCacheSize As Integer)
   Dim StartTime, EndTime
   Dim rs As ADODB.Recordset

   Set rs = New ADODB.Recordset
   rs.CacheSize = iCacheSize

   StartTime = Timer
   rs.Open "SELECT * FROM customers", cn
   While Not rs.EOF
      rs.MoveNext
   Wend
```

```
    EndTime = Timer

    MsgBox "Time for CacheSize = " & iCacheSize & " is " & _
        EndTime - StartTime & " seconds. " & _
        rs.RecordCount & " records"
End Sub
```

ملاحظة إذا قمت باستخدام ذاكرة وسيطة لسجل ذات مجموعة سجلات للأمام فقط، من الممكن الانتقال إلى الخلف في مجموعة السجلات حتى حدد السجلات المخزنة. لا يفضل إجراء ذلك لأنه يعني أن قد يجب إعادة تنفيذ الاستعلام.

إحضار الصفوف المتعددة

كما رأينا مع RDO وDAO، يمكننا استخدام أسلوب GetRows الخاص بمجموعة السجلات لإحضار عدداً كبيراً من الصفوف في مصفوفة متباينة (انظر الفصل ١٤ "وحدات الوصول إلى البيانات" للإطلاع على مزيد من التفاصيل عن المصفوفات المتباينة). في ADO، اصبح أسلوب GetRows أكثر مرونة وفعالية شيئاً ما. صيغة الأسلوب هي كالتالي:

```
variant_array = rs.GetRows(Rows, Start, Fields)
```

المعاملات الاختيارية للأسلوب هي:

Rows: يوضح المعامل عدد الصفوف التي يجب استرجاعها. الافتراض هو **adGetRowsRest**: الذي يحضر كل الصفوف المتاحة الباقية.

Start: يوضح هذا المعامل إشارة مرجعية إلى المكان الذي يجب أن يبدأ عنده استرجاع السجل.

Fields: يعتبر هذا المعامل خاصية جديدة تتيح لك تحديد متغير متباين يحتوي على تفاصيل حقل أو أكثر تريد استرجاعه. يمكنك تحديد إما أسماء الحقول أو مواضع الحقول الترتيبية.

ملاحظة تبدو إمكانية تحديد الحقول التي تريدها بواسطة أسلوب GetRows فكرة جيدة، ولكنها إجراءاً ضعيفاً على أن يتم استخدامه فعلياً. يجب أن تقوم دائماً بتحديد الحقول التي تريدها فقط في جملة SELECT. دع قاعدة بيانات أوراكل تقوم بما تستطيع القيام به من تصفية.

توضح التعليمات البرمجية رقم ١٧-٨ مثالاً لاستخدام أسلوب GetRows الخاص بـ ADO. لاحظ أنك تحتاج فقط لتعريف المصفوفة المتباينة كـ Variant. ستهتم ADO بضمان أن المصفوفة بالحجم الصحيح. يحدد المثال قيمتين لمعاملات GetRows، ولكن تركهما لأنهما افتراضيين على أي حال. أخيراً، لاحظ كيف تستخدم وظيفة Ubound لتحديد عدد السجلات والحقول في المصفوفة المتباينة وكيف أنها تقوم على صفر، لذا يجب إضافة ١ لكل قيمة من هذه القيم.

تعليمات برمجة ١٧-٨

```
Private Sub GetRsRows()
' This procedure is an example of using GetRows.
    Dim rs As ADODB.Recordset
    Dim vArray As Variant

    Set rs = New ADODB.Recordset
rs.Open "SELECT * FROM movies", cn, adOpenForwardOnly, _
            adLockReadOnly
    vArray = rs.GetRows(adGetRowsRest, rs.Bookmark)
    MsgBox "There are " & UBound(vArray, 2) + 1 & " records" & _
        " and " & UBound(vArray, 1) + 1 & " fields."
```

تغيير الصف الحالي

إن ADO تشبه DAO وRDO كثيراً في أن لديها أساليب Move الأربعة التي تتوقعها: MoveFirst وMovePrevious وMoveNext وMoveLast. كما أن لديها أسلوباً إضافياً وهو Move، صيغته كالتالي:

```
rs.Move NumRecords, Start
```

NumRecords هي عدد السجلات التي تريد نقلها، وقد تكون القيمة موجبة للاتجاه الأمامي وقد تكون سالبة للاتجاه الخلفي. توفر Start الفرصة للإشارة إلى مكان البداية. الافتراض هو الاتجاه الحالي "adBookmarkCurrent"، بالرغم من أنه بإمكانك أيضاً اختيار adBookmarkFirst أو adBookmarkLast لكي تبدأ عند بداية مجموعة السجلات أو عند النهاية.

مثلما هو الأمر مع أساليب الوصول للبيانات الأخرى، يجب فحص خصائص BOF و EOF الخاصة بمجموعة السجلات لكي تضمن أنك تتصل بسجل صحيح. إذا كانت كلتا الخاصيتين صحيحتين، إذن فلديك مجموعة سجلات فارغة.

الوصول إلى حقول مجموعة السجلات

كما هو الحال مع DAO، هناك ثلاثة طرق بسيطة للإشارة إلى حقل في مجموعة سجلات rs بمجرد قيامك بتكوينه، إما بالأرقام الترتيبية أو بالاسم أو بصيغة!.

```
rs(1)
rs("movie_name")
rs!movie_name
```

تحذير كن حذراً عند استخدام حقل مثل (1)rs. يتم تخزين الحقول بنفس طريقة تخزينها في جدول الأساس ٢ وليس في جملة SELECT، كما هو الحال في كل أسلوب آخر للوصول للبيانات مرة أخرى.

إذا كان هناك خطراً من الحقل الذي يحتوي على قائمة NULL (خالي)، يجب إذن ضم سلسلة طولها صفر إلى الحقل عند تخصيصها لمتغير.

توضح التعليمات البرمجية رقم ١٧-٩ إجراء FillListBox مرة أخرى، يتم كتابة هذه المرة بالرغم من أنه بإمكاننا تحديد هذه الخصائص ظاهرياً بالمعاملات الصحيحة في أسلوب Open الخاص بمجموعة السجلات.

تعليمات برمجة ١٧-٩

```
Public Sub FillListBox(ListBox As ComboBox)
' Fill a list box with movie names
  Dim rs As ADODB.Recordset
  Set rs = New ADODB.Recordset

  rs.Open "SELECT * FROM movies", cn
  ListBox.Clear
  With rs
```

```
      While Not .EOF
         ListBox.AddItem rs("movie_name")
         .MoveNext
      Wend
   ListBox.ListIndex = 0  ' Display the first item
   End With
   Set rs = Nothing
End Sub
```

تغيير الصفوف والقيم في مجموعة سجلات

إذا كنت مستخدماً مجموعة سجلات قابلة للتحديث، إذن عندما تنتقل إلى الصف الذي تريد تغييره، يمكنك إعداد القيم بنفس الطريقة التي رأيناها مع DAO و RDO. يعتبر تحديث سجل موجود أمراً مختلفاً قليلاً في ADO عن التقنيات الأخرى، وإنها حقاً صعبة الترميز حيث أن أسلوب Edit لم يعد موجوداً. تهتم ADOآلياً بذلك من أجلنا وتحافظ على حالة التحرير في خاصية Edit Mode الخاصة بمجموعة السجلات. (ستتذكر من الأمثلة DAO وRDO أنه تم الحفاظ على هذه الحالة في تعليمات البرمجة ظاهرياً). يمكن أن تكون لخاصية Edit Mode إحدى القيم الثلاثة كما هو موضح في الجدول ١٧-٧.

الجدول ١٧-٧: القيم الممكنة لخاصية Edit Mode الخاصة بمجموعة السجلات.

الثابت	البيان
AdEditNone	يدل هذا الثابت على أنه لا يوجد تحرير تحت التشغيل.
AdEditInProgress	يوضح هذا الثابت أن البيانات الموجودة في السجل الحالي تم تغييرها ولكن لم يتم حفظها.
AdEditAdd	يوضع هذا الثابت أنك قد قمت باستخدام أسلوب Add New، وأن السجل لم يتم حفظه.

تعني هذه الخاصية الجديدة أنه يمكنك الاستمرار وإجراء تغييرات لسجل دون استخدام أسلوب Edit. (عدم ظهور هذه التقنية عند إصدار RDO يعتبر أمراً غريباً). بعد إجراء التغييرات، يمكنك استخدام أسلوب Update لحفظ التغييرات على قاعدة البيانات.

بالرغم من أنك تستطيع استخدام كلاً من أسلوب Add New لإضافة سجلاً جديداً لمجموعة السجلات Update لحفظه على قاعدة البيانات، هذان الأسلوبان لديهما صيغة موسعة. يسمح لك هذان الأسلوبان بتضمين قاعدة قيم وأسماء الحقول لاستدعاءات الأسلوب.

للإطلاع على مثال لاستخدام هذه الأساليب، انظر قسم "Putting It All Together," التابع لهذا الفصل.

البحث عن صف معين

مرة أخرى، مثلما هو الأمر مع RDO، لا يوجد أساليب Find "بحث" في ADO. هناك طريقة أفضل لأداء نفس المهمة وهي تضييق عبارة WHERE عند تكوين مجموعة السجلات.

إغلاق مجموعة سجلات

كما هو الحال مع كافة تقنيات الوصول للبيانات، إذا كان هدفك هو بنية تطبيق قابلة للتحجيم، يجب محاولة إغلاق أي مجموعات سجلات بأقصى سرعة. لإجراء ذلك، يمكنك استخدام أسلوب Close على مجموعة السجلات أو جعله يساوي Nothing لإزالة الوحدة تماماً من الذاكرة. الأسلوب الأخير هو الأسلوب المفضل إذا كنت ستقوم بإعادة استخدام مجموعة السجلات.

أحداث مجموعة السجلات

مثلما تتوقع في أسلوب وجدول للبيانات يزودك بنموذج تتحكم فيه الأحداث بصورة كبيرة، لا تختلف وحدة Recordset عن ذلك وتزودك بمجموعة متنوعة كبيرة من الأحداث (انظر الجدول ١٧-٨). تكون هذه الأحداث مفيدة جداً عند استخدام مؤشر قابل للتحديث.

الجدول ١٧-٨: الأحداث المتاحة لمجموعة السجلات.

اسم الحدث	البيان
EndOfRecordset	قمت بمحاولة تعدي السجل الأخير.
FetchComplete	إتمام فتح مجموعة سجلات غير متزامنة.
FetchProgress	يخبر هذا الحدث بتشغيل عملية فتح سجلات غير متزامنة.
FieldChangeComplete	تم تغيير قيمة حقل.

الجدول ١٧-٨: الأحداث المتاحة لمجموعة السجلات.

اسم الحدث	البيان
MoveComplete	تغيير السجل الحالي.
RecordChangeComplete	تم تغيير سجل أو سجلات وحفظها على قاعدة البيانات (كما هو الحال مع أسلوب AddNew أو Update method).
RecordsetChangeComplete	تم تغيير مجموعة السجلات (كما هو الحال مع أسلوب Resync أو Close).
WillChangeField	قيمة شغل على وشك التغيير.
WillChangeRecord	سجل واحد أو أكثر على وشك التغيير والحفظ على قاعدة البيانات.
WillChangeRecordset	مجموعة السجلات على وشك التغيير.
WillMove	السجل الحالي على وشك التغيير.

تلميح كما ترى الأحداث المسرودة في الجدول ١٧-٨، يمكنك ترميز عدداً كبيراً من الأحداث يستطيع أيضاً توفير قدر كبير من المعلومات عن العارض الذي يسبب إطلاق الحدث. يمكن أن تكون بعض هذه الأحداث مفيدة، على سبيل المثال، يمكن استخدام WillChangeField لترميز البرامج الفرعية لمراجعة البيانات، ويمكن استخدام EndOfRecordset لإضافة سجل جديد إذا تعديت نهاية مجموعة السجلات.

على أي حال، يجب أن تكون متيقظاً للقوى التي تمنحها لك هذه الأحداث وتحاول استخدامها فقط لحل بعض المشاكل الصعبة الأخرى. في حالة استخدام الأحداث كثيراً، سيصبح تطبيقك معقداً وصعب الحفاظ عليه. ضع في اعتبارك أن بسبب قيام نموذج ADO بتوفير الوصول لأي شيء تفكر فيه، ليس عليك استخدام كل هذه القوى في كل فرصة.

وحدة Command "الأمر"

مع أسلوب Open الذي قمنا باستخدامه لتكوين مجموعة سجلات، مررنا بسلسلة تحديد جملة SELECT التي نريد تشغيلها. لم يكن هناك شيئاً آخر نستطيع عمله معها، لم نستطيع، على سبيل المثال، استخدام معاملات كما فعلنا مع استعلامات rdo عند استخدام وحدات البيانات البعيدة (RDO). للحصول على هذه الوظيفة الإضافية وتقنيات الترميز المطورة التي تأتي معها، يجب أن تقوم باستخدام وحدة ADO الرئيسية الثالثة وهي وحدة Command.

توجد وحدة Command أسفل وحدة Connection في تدرج ADO، بالرغم من أننا نستطيع مرة أخرى تكوين أمراً منفرد وإقرانه باتصال عندما تريده أن يعمل شيئاً. في الواقع، حيث لا يوجد مثيل لأسلوب rdoCreateQuery، ليس لدينا خيار إلا تكوين وحدات Commandمنفردة ثم تعيين خصائصها. أول شيء يجب النظر إليه هو الخصائص التي يمكنك استخدامها لوحدة Command. لحسن الحظ لا يوجد العديد منها (انظر الجدول ١٧-٩).

الجدول ١٧-٩: الخصائص الممكنة لوحدة Command.

الخاصية	البيان
ActiveConnection0	توضح هذه الخاصية الاتصال الذي تريد استخدامه للأمر أو سلسلة تحدد اتصالاً جديداً.
CommandText	توضح هذه الخاصية نص الأمر، مثل جملة DML أو إجراء المخزن.
CommandTimeout	تظهر هذه الخاصية عدد الثواني قبل انتهاء وقت الأمر (العدد الافتراضي هو ٣٠).
CommandType	توضح هذه الخاصية نوع الجملة التي تمثلها خاصية CommandText. انظر الجدول ١٧-١٠ للإطلاع على قائمة من القيم الممكنة.
Prepared	تحدد هذه الخاصية ما إذا كانت الجملة في خاصية CommandText تحميلها مزود البينات بشكل مجمع. لا يتناسب ذلك مع مستخدمي أوراكل.

الجدول ١٧-٩: الخصائص الممكنة لوحدة Command.

الخاصية	البيان
State0	تشير هذه الخاصية إلى حالة وحدة Command. تعتبر هذه الخاصية مفيدة اكثر لمجموعات السجلات، ولكن يمكنها، على سبيل المثال، تضمين adStateExecuting إذا كنت قد قمت بتنفيذ الأمر بصورة غير متزامنة. قد تكون قيمة هذه الخاصية هي مجموع قيم عديدة.

الجدول ١٧-١٠: الثوابت الممكنة لخاصية CommandType الخاص بوحدة Command

الثابت	البيان
AdCmdText	تحتوي CommandText على النص الكامل للاستعلام.
AdCmdTable	CommandText هو اسم جدول قم رجوع أعمدته كافة. (يتم تكوين SQL من أجلك).
AdCmdStoredProc	CommandText هو اسم إجراء مخزن.
AdCmdUnknown	نوع الأمر في خاصية CommandText غير معروفاً (الافتراض) قد يكون على ADO استجواب قاعدة البيانات لملء المعلومات المفقودة.

تحذير: لا تستخدم أي أمر يستخدم adCmdTable لأنه يقوم بإرجاع الجدول بأكمله. معظم جداول قواعد بيانات أوراكل تكون كبيرة جداً حتى تكون عملية ومناسبة.

تسهل ADO استخدام أي خاصية من خصائصها، وذلك عن طريق توفير أي معلومات لم تقم بتوفيرها. على سبيل المثال، إذا لم تحدد نوع خاصية CommandText التي تستخدمها، تستطيع ADO معرفة ذلك. على أي حال، تفيض عليها القيام برحلة إلى خادم البيانات في كل مرة تحتاج

إلى العثور على هذه المعلومات المفقودة. من الأفضل توفير كافة المعلومات في وقت التصميم. لذا دعنا نفعل ذلك الآن. توضح التعليمات البرمجية مثالاً لإجراء جديد يسمى CreateCommands الذي يقوم بإعداد أي وحدات Command في الوحدة النمطية الحالية. يمكنك استدعاء هذا الإجراء من حدث Form_Load.

تعليمات برمجة ١٧-١٠

```
Private cmd As New ADODB.Command
Private Sub CreateCommands()
   Dim rs As ADODB.Recordset
   With cmd
      .ActiveConnection = cn
      .CommandType = adCmdText
      .CommandText = _
"SELECT * FROM video_store.movies WHERE rating = 'G'"
   End With
' Test it out for the sake of the example.
   Set rs = cmd.Execute
   MsgBox rs(1)
End Sub
```

يقدم المثال في التعليمات البرمجية رقم ١٧-١٠ الاتصال ونص الأمر المقرر تنفيذه ونوع الأمر. في هذه الحالة، يقوم الأمر بتنفيذ جملة SELECT، لذلك نستخدم خاصية CommandType الخاصة بـ adCmdText. لاحظ أنك تستطيع بسهولة استخدام اسم مخطط قاعدة البيانات مع الاستعلام. على أي حال، لا يعتبر هذا المثال مثيراً للغاية حيث أننا لا نستطيع تغيير الاستعلام بأي طريقة، بالرغم من أننا نستطيع تغيير إحدى "أو أكثر" خصائص الأمر وتحديد مجموعة السجلات بسهولة بواسطة أسلوب Execute.

معاملات الأمر

هناك أسلوبان فقط لوحدة command، وقد أوضح لنا القسم السابق أحدهما وهو أسلوب Execute. يتيح لنا الأسلوب الثاني إضافة معامل إلى مجموعة في كل من ODBCDirect وRDO، وتتيح لنا هذه المعاملات تخزين قيمة سيتم إدراجها في الاستعلام أو استدعاء الإجراء المخزن في وقت تنفيذ الأمر. على سبيل المثال، يمكننا إعداد استعلام يستخدم معامل، ثم تجديد

مجموعة السجلات "بأسلوب Resync" تكراراً مع تغيير قيمة أو قيم المعامل في كل مرة نقوم بتنفيذ الأمر.

قبل التعرف على كيفية استخدام المعاملات، يجب أولاً أن نقوم بتكوينها. نحن نفعل ذلك بأسلوب CreateParameter الخاص بوحدة Command. يؤدي هذا الأسلوب إلى تكوين مجموعة مرتبة من المعاملات. صيغة أسلوب CreateParameter هي:

```
Set parameter = command.CreateParameter (Name, Type, Direction,
Size, Value)
```

المتغيرات الاختيارية الخاصة بالأسلوب:

Name: يشير هذا المتغير إلى اسم وحدة parameter.

Type: يشير هذا المعامل إلى نوع بيانات المعامل. انظر خاصية Type في ملف التعليمات الخاص بفيجوال بيسك للإطلاع على قائمة كاملة من أنواع البيانات التي تريد العمل بها في أوراكل موضحة في الجدول ١٧-١١.

Direction: يشير هذا المتغير إلى اتجاه المعامل. انظر الجدول ١٧-١٢ للإطلاع على إعدادات صالحة بالإضافة إلى كيفية تناسبها مع طور PL/SQL الخاص بأوراكل.

Size: يشير هذا المتغير إلى أقصى طول لقيمة المعامل بالأحرف أو البايت.

Value: يشير هذا المتغير إلى قيمة المعامل. ستقوم عادةً بتقديم هذه القيمة قبل تنفيذ الأمر مباشرة.

لاحظ أن جميع هذه المتغيرات اختيارية لأنك تستطيع إما تزويدها بالمعامل (أنظر الجدول ١٧-١٣) أو في بعض الأحوال، تركها.

الجدول ١٧-١١: أنواع البيانات المستخدمة مع قاعدة بيانات أوراكل نموذجياً.

نوع بيانات أوراكل الأساسي	نوع بيانات معامل ADO
DATE	AdDate
LONG	AdLongVarChar
VARCHAR2	AdVarChar
NUMERIC	AdNumeric
RAW	AdVarBinary
VARCHAR2	AdVarChar

الجدول ١٧-١٢: متغير الاتجاه الخاص بوحدة CreateParameter.

الثابت	طور PL/SQL	البيان
AdParamUnknown		يشير هذا الثابت إلى أن الاتجاه المعامل غير معروف من أجل تحقيق كفاءة عالية، يجب عدم استخدام هذا الثابت.
AdParamInput "افتراضي"	IN	يشر هذا الثابت إلى معامل مدخلات.
AdParamOutput	Out	يشير هذا الثابت إلى معامل مخرجات.
AdParamInputOutput	IN OUT	يشير هذا الثابت إلى أن المعامل يكون لكل من المدخلات والمخرجات.
AdParamReturnValue	OUT	يشير هذا الثابت إلى قيمة رجوع.

الجدول ١٧-١٣: خصائص معامل الأمر

الخاصية	البيان
السمات	يحدد هذه الخاصية ما إذا كان المعامل يمكن أن يحتوي على أرقام موقعة وقيم خالية وبيانات RAW.
الاتجاه	تحدد هذه الخاصية اتجاه المعامل (مدخلات أو مخرجات أو كلاهما) كما هو موضح في الجدول ١٧-١٢.
الاسم	تحدد هذه الخاصية اسماً عشوائياً للمعامل يمكن أن تكون سلسلة طولها صفر.
NumericScale "مقياس عددي"	تحدد هذه الخاصية عدد المواضع العشرية لمعامل عددي.
الدقة	تحدد هذه الخاصية أكبر أرقام في معامل عددي.

الجدول ١٧-١٣: خصائص معامل الأمر

الخاصية	البيان
الحجم	تحدد هذه الخاصية أقصى طول لمعامل متغير الطول، مثل adVarChar.
النوع	تحدد هذه الخاصية نوع بيانات المعامل الذي رأيناه في الجدول ١٧-١١.
القيمة	تحدد هذه الخاصية قيمة المعامل. يمكن تغيير ذلك ويتم تنفيذ الأمر مرة أخرى.

الآن بعد ما اطلعنا على كل المواصفات التقنية لوحدات Parameter، حان الوقت لتجميعها سوياً. نحن نتعامل مع استعلام SELECT بمعاملات في هذا القسم وننتقل لاستخدامات أخرى للمعاملات في الأقسام القليلة التالية:

عندما تتعامل مع استعلامات كما رأينا في القسم السابق، يمكنك تكوين وحدة Command بالطريقة التالية:

```
With cmd
    .ActiveConnection = cn
    .CommandType = adCmdText
    .CommandText = _
        "SELECT * FROM video_store.movies WHERE rating = 'G'"
End With
```

كما هو الحال مع ODBCDirect و RDO، يمكنك استخدام معاملات في هذا الاستعلام بدلاً من ترميز قيمة "G"> leglh I, hGlv td Hshgdf ODBC"، يجب استخدام نواب لمواضع المعاملات لتحديد المكان الذي يجب أن يكون فيه الاستعلام، ومثلما سبق، نستخدم علامة الاستفهام.

نقوم بتجميع ذلك في مثال في التعليمات البرمجية رقم ١٧-١١، يحتوي هذا المثال على إجراء CreateQueryCommand الذي سيتم استدعاءه في حدث Form_Load أو حدث Initialization الخاص بـ Class. ولأننا نريد أن يكون الأمر والمعاملات الخاصة به متاحة خلال الوحدة النمطية، نقوم بتعريفهم على مستوى الوحدة النمطية. في الإجراء، لدينا وسائل

عديدة مختلفة لترميز نفس الشيء. ولأن وحدة command ليس لديها أحداث خاصة بها، ليـــس هناك حاجة لاستخدام كلمة الأساس WithEvents، لذلك نستخدم كلمة الأساس New في تعريـف Dim.

تعليمات برمجة ١٧-١١

```
Private qy1 As New ADODB.Command
Private pYear As New ADODB.Parameter
Private pRating As New ADODB.Parameter
Private Sub CreateQueryCommand()
    Dim qy1 As New ADODB.Command
    Dim strSQL As String

    Dim pYear As New ADODB.Parameter
    Dim pRating As New ADODB.Parameter

    strSQL = "SELECT * FROM movies WHERE year_of_release > ? "
    strSQL = strSQL & " AND rating = ? ORDER BY movie_name"

 With pYear
        .Direction = adParamInput
        .Type = adNumeric
    End With

    With pRating
        .Direction = adParamInput
        .Type = adVarChar
        .Size = 6
    End With

    With qy1
        .CommandText = strSQL
        .CommandType = adCmdText
```

```
        .ActiveConnection = cn
        .Parameters.Append pYear
        .Parameters.Append pRating
    End With

End Sub
```

إعداد جملة SELECT في سلسلة بها نواب مواقع المعامل المتضمنة، نقوم بإعداد تفاصيل المعاملين المطلوبين: pYear وpRating. كان بإمكانك استخدام أسلوب CreateParameter هنا، ولكن ADO سيعالج ذلك آلياً. يجب توفير معلومات كافية حتى تصبح المعاملات صالحة، بالرغم من أنك لن تتلقى أي رسائل خطأ أن تحاول إلحاق المعامل الأمر. يجب أن يقوم باستخدام صيغة WITH...END WITH عندما يكون لديك أكثر من خاصية نقوم بتعينها، لأنها أسرع ويمكن قراءتها بسهولة أكثر.

الإجراء الأخير في التعليمات البرمجية رقم ١٧-١١ هو إعداد الأمر نفسه حتى هذه النقطة، يعتبر وحدة منفردة. نقوم الآن بإضافة المعلومات الهامة، مثل ActiveConnection التي سيقوم الأمر بالعمل عليها ثم إلحاق المعاملات.

بمجرد قيامك بتكوين أمر مثل هذا، تحتاج لتنفيذه حتى تتأكد أنه يعمل جيداً. توضح التعليقات البرمجية رقم ١٧-١٢ إجراء TestQueryCommand الذي لا يقوم فقط باختيار الأمر ولكنه يوضح أيضاً كيفية تشغيله بصورة طبيعية. أنك توفر قيم لكافة المعاملات ثم تقوم.بتنفيذه لإرجاع مجموعة سجلات. تذكر أن المعاملات يتم تحديدها على مستوى الوحدة النمطية.

تعليمات برمجة ١٧-١٢

```
Private Sub TestQueryCommand()
    Dim rs As New ADODB.Recordset
    pRating = "PG-13"
    pYear = 1992

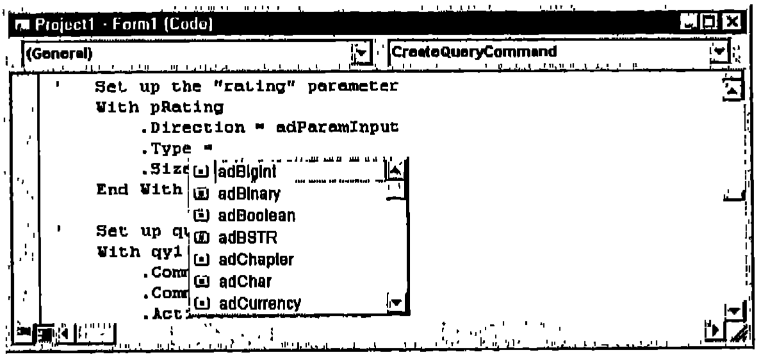
    Set rs = qy1.Execute

    While Not rs.EOF
```

```
      MsgBox rs(1)
      rs.MoveNext
    Wend
    Set rs = Nothing
End Sub
```

ملاحظة يتم قصر مجموعة السجلات التي تم إرجاعها من أمر Execute على مؤشر ثابت للأمام فقط.

عندما تقوم بإصدار تعليمات البرمجة، يقدم لك فيجوال بيسك قدر كبير من المساعدة. على سبيل المثال، في الشكل ١٧-٥، يمكنك أن ترى أن ليس عليك تذكر كلمة الأساس التي يجب استخدامها لنوع البيانات. تقوم قائمة منسدلة بتوفير ذلك لك.



الشكل ١٧-٥ استخدام خصائص IntelliSense الخاصة بفيجوال بيسك لتبسيط الترميز.

بمجرد العمل مع المعاملات لفترة معينة، ستدرك أن هناك طرق لا حصر لها لترميز إجراء مثل ذلك الشيء الرئيسي هو أن عندما تجد نمطاً يعجبك، التزم بهذا النمط بهدف تطوير الشكل وإمكانية الصيانة. يتم توضيح نسخه مكثفة أكثر من التعليمات البرمجية رقم ١٧-١١ في التعليمات البرمجية رقم ١٧-١٣. إنها تفعل نفس الشيء ولكنها تستخدم أسلوب CreateParameter لتكوين معامل، وأسلوب Append لإضافته للمجموعة في سطر واحد. لتنفيذ الأمر، ستحتاج لاستخدام إجراء مثل ذلك الموضح في التعليمات البرمجية رقم ١٤-١٧. ولأن المعاملات لم يتم تكوينها كوحدات، يجب الإشارة إليها من خلال مواضعها في مجموعة المعاملات.

تعليمات برمجة ١٧-١٣

```
Private qy2 As New ADODB.Command
Private Sub CreateQueryCommand2()
    Dim strSQL As String

    strSQL = "SELECT * FROM movies WHERE year_of_release > ? "
    strSQL = strSQL & " AND rating = ? ORDER BY movie_name"

    With qy2
        .CommandText = strSQL
        .CommandType = adCmdText
        .ActiveConnection = cn
        .Parameters.Append .CreateParameter _
            (, adNumeric, adParamInput)
        .Parameters.Append .CreateParameter _
            (, adVarChar, adParamInput, 6)
    End With

End Sub
```

تعليمات برمجة ١٧-١٤

```
Private Sub TestQueryCommand2()
    Dim rs As New ADODB.Recordset
    qy2(1) = "PG-13"       ' Rating
    qy2(0) = 1992          ' Year
Set rs = qy2.Execute   ' Create the Recordset
End Sub
```

تنفيذ أوامر SQL

ستحتاج في كثير من الأوقات إلى تنفيذ أوامر SQL. ستحتاج، بصورة طبيعية إلى تنفيذ أوامر DML وهي INSERT وUPDATE وDELETE، ولكنك غير محدد بهذه الأوامر الثلاثة فقط.

حيث أن مجموعات السجلات القابلة للتحديث لا تعتبر تصميماً قابلاً للتدرج، ستتم كافة التحديثات بواسطة SQL أو كما سنرى في القسم التالي، بواسطة الإجراءات المخزنة (وهذا ما هو أفضل).

لتنفيذ أوامر SQL البسيطة، تستخدم وحدة Connection الخاصة بـ ADO. يقوم المثال بحذف صف من جدول Movies. لاحظ حدوث تثبيت ضمني على الأمر كما هو الحال مع RDO. إذا أردت أن يكون ذلك جزءاً من حركة، فسيكون عليك استخدام أساليب BeginTrans/CommitTrans.

تعليمات برمجية ١٧-١٥

```
Dim strSQL As String
strSQL = "DELETE FROM movies WHERE movie_id = 303"
cn.Execute strSQL
```

يعتبر استخدام وحدة Connection لتنفيذ أوامر SQL أمراً معقولاً، ولكننا نحتاج لاستخدام وحدة Connection للاستفادة من قوة الأوامر ذات المعاملات. يمكننا استخدام نفس الصيغة التي رأيناها في القسم السابق (انظر التعليمات البرمجية رقم ١٧-١٦). هناك، الآن اختلافاً صغيراً ولكنه رئيسياً وهو أن الأمر لا يقوم بإرجاع مجموعة سجلات. نحن لا نقوم بإعداد وحدة Recordset، إننا نقوم بمجرد تنفيذ الأمر كما ترى في التعليمات البرمجية رقم ١٧-١٧.

تعليمات برمجيه ١٧-١٦

```
Private Sub CreateQueryCommand3()
   Dim strSQL As String

   strSQL = "DELETE FROM movies WHERE movie_id = ?"
With qy3
      .CommandText = strSQL
      .CommandType = adCmdText
      .ActiveConnection = cn
      .Parameters.Append .CreateParameter _
         (, adNumeric, adParamInput)
   End With
End Sub
```

تلميح يستخدم الإجراء TestQueryCommand3، الموجـــود فـــي التعليمـــات البرمجيـــة رقـــم ١٧-٧، أســـاليب BeginTrans و CommitTrans الخاصة بوحدة Connection للتحكم في تثبيـت التغيـيرات فـي قـاعدة البيانات. في حالة وجـود تغيـيرات كثـيرة داخـل الحركـة، سـتلاحظ تطور مميزاً في الأداء، ولكن ســيكون ذلـك علـى حسـاب الصفـوف التي يتم أقفالها للفترة الزمنية الخاصة بحركـــة أطـول.

تعليمات برمجية ١٧-١٧

```
Private Sub TestQueryCommand3()
    cn.BeginTrans
    qy3(0) = 71          ' Movie_id
    qy3.Execute          ' Delete the movie
    cn.CommitTrans

End Sub
```

استدعاء الإجراءات المخزنة

تسمح لك أساليب الوصول للبيانات التي يمكنك استخدامها من فيجوال بيسك بتنفيـذ الإجـراءات المخزنة، كما تسمح لك كل هذه الأساليب عدا DAO/JETبإرجاع معاملات المخرجات. يمكنـــك تنفيذ إجراءات مخزنة بصيغة Begin/End الخاصة بقطع PL/SQL غير مسماه التي رأيناها فـي الفصول السابقة. هناك طريقة أفضل وهـي اسـتخدام وحـدة Command لكـل اسـتدعاءات الإجراءات.

كما رأينـــا فـــي الجـداول ١٧-٩ و١٧-١٠، يمكننـا تحديـد CommandType لوحـدة Command كـ adCmdStoredProc. يعني ذلك أن في خاصيـــة CommandText يجـب فقط إدخال اسم الإجراء المخزن ولا يجب أن تقوم بتضمينه في جمـــل Begin/End ولا يجـب أيضاً أن تقوم بإدخال سلسلة نص بنواب لمواقع معاملات. من ناحية أخرى، يجب أن تتأكد مـــن أن كل المعاملات يتم إلحاقها بصورة صحيحة بمجموعة وبالترتيب الصحيح.

كمثال لكيفية تحقيق ذلك، سنستخدم الإجراء المخزن movies_insert3 الذي رأيناه في الفصـل ١٥ "وحدات البيانات البعيدة" في التعليمات البرمجية رقم ١٥-١٣. يختلف إعداد وتنفيذ الإجراء المخزن

في ADO مقارنة بـ RDO. توضح لك التعليمات البرمجية رقم ١٧-١٨ كيفية إعداد الأمـر. يعتـبر العيب الوحيد للتقنيات التي يستخدمها ADO هو أن لدينا كمية أساسية من الرموز لنقوم بتكوينها مع كل إجراء مخزن نريد أن نقوم بتنفيذه. سنرى في هذا القسم "استخدام بيئة البيانات" كيف نجعل هذه الوظيفة بأكملها أكثر سهولة.

تعليمات برمجة ١٧-١٨

```
Private Sub CreateQueryCommand4()
   Dim strSQL As String

   With qy4
      .CommandText = "video_store.movies_insert3"
      .CommandType = adCmdStoredProc
      .ActiveConnection = cn
      .Parameters.Append .CreateParameter _
         (, adNumeric, adParamInputOutput)
      .Parameters.Append .CreateParameter _
         (, adVarChar, adParamInput, 50)
      .Parameters.Append .CreateParameter _
         (, adNumeric, adParamInput)
      .Parameters.Append .CreateParameter _
         (, adVarChar, adParamInput, 6)
      .Parameters.Append .CreateParameter _
         (, adNumeric, adParamInput)
   End With

End Sub
```

توضح التعليمات البرمجية رقم ١٧-١٩ الطريقة التي يمكننا اتباعها لتنفيذ الإجراء المخـزن الذي تم إعداده كأمر في التعليمات البرمجية رقم ١٧-١٨. ولأننا لم نقم بتكوين معاملات منفردة، يجب أن نشير إليها مرة أخرى من خلال مكانها في مجموعة المعاملات لكل من قيم المدخـلات وقيم المخرجات.

تعليمات برمجية ١٧-١٩

```
Private Sub TestQueryCommand4()
    qy4(0) = 0          ' Movie_id
    qy4(1) = "A short about shorts"
    qy4(2) = 33
    qy4(3) = "G"
    qy4(4) = 1999
    qy4.Execute
    MsgBox "The id used for the movie was " & qy4(0)
End Sub
```

استدعاء وظائف PL/SQL

مرة أخرى، لا يمكننا ترك الإجراءات المخزنة دون التعرف على كيفية استدعاء وظائف PL/SQL. مثلما فعلنا مع RDO، سنقوم بإصدار بعض الرموز التي تستدعي الوظيفة GET_MOVIE_ID التي رأيناها في التعليمات البرمجية ١٥-١٦. لا يوجد فرق كبير بين استدعاء وظيفة وإجراء مخزن، الشيء الوحيد الذي يجب أن تتذكره هو أن المعامل الأول في المجموعة يجب أن يكون قيمة رجوع الوظيفة، ويجب أن تقوم بالترميز بنوع adParamReturnValue. يعمل أي شيء آخر على النحو الذي تتوقعه (انظر التعليمات البرمجية رقم ١٧-٢٠).

تعليمات برمجية ١٧-٢٠

```
Private Sub TestFunctionCall()
    Dim qy As New ADODB.Command
    With qy
        .CommandText = "video_store.get_movie_id"
        .CommandType = adCmdStoredProc
        .ActiveConnection = cn
        .Parameters.Append .CreateParameter _
            (, adNumeric, adParamReturnValue)
        .Parameters.Append .CreateParameter _
            (, adVarChar, adParamInput, 50)
```

```
    End With
qy(1) = "Awakenings"
qy.Execute
MsgBox "The id for the movie is " & qy(0)
End Sub
```

استرجاع جداول PL/SQL

في الفصل الخامس عشر، ذكرنا أنك تستطيع استرجاع جداول PL/SQL دون التوجه إلى وحدات أوراكل (OLE)، لذلك إذا كنت ستستخدم تقنية مايكروسوفت، يجب الاعتياد على هـــذه الطريقـــة. يعتبر تحويل جدول PL/SQL إلى مجموعة سجلات أمر غير معتاد ولكن في الوقت الحالي، لـــم تعد هذه الخاصية متاحة خلال مزود OLE-DB لأوراكل.

لأن التقنية التي يجب استخدامها مختلفة عن إجراء مخزن عادي أو استدعاء وظيفة، ولأنـــها تحتاج لاستخدام حزمة PL/SQL، لا يمكننا استخدام الصيغة سهلة الاستخدام التي رأيناها في قسم "استدعاء الإجراءات المخزنة"، بدلاً من ذلك، يجب الرجوع إلى صيغة نـــائب الموقـــع وجملـــة الاستدعاء التي رأيناها في الفصل ١٥. يعتبر تحديد سلسلة الأوامر هي أهم جزء مـــن التقنيـــة. عليك شكلان مختلفان للصيغة:

```
{call package.proc ({RESULTSET max_recs, table1, table2 ...})}
{call package.proc({RESULTSET max_recs, table1}, { RESULTSET
max_recs, table2},...)}
```

تحتاج كلا الشكلين لاهتمام. يعتبر الشكل الأول أبسط من حيث تعريفه كما أنه الأقوى. تقـــوم تعليمات البرمجة باستدعاء إجراء محزم (packaged procedure) يحتـــوي علـــى جـــدول أو جداول PL/SQL وجدول ١ مسمى وجدول PL/SQL، كما ذكرنـــا فـــي الفصـــل ١٠ "تكـــوين الإجراءات المُخزنة" هو عبارة عن مجموعة بسيطة حقاً.أما بالنسبة لفيجوال بيسك، يعتبر جـــدول PL/SQL عندما يصل جدول PL/SQL لفجيوال بيسك، يصبح مصفوفة على أساس صفـــر (أي يبدأ الفهرس عند صفر). تقوم التقنية الأولى بدمج أي عدد من جداول PL/SQL فـــي مجموعـــة سجلات واحدة ويضع عناصر من كل جدول في حقول متتالية من مجموعة السجلات. أنك تحتاج للتأكد من أن كل الجداول لديها نفس عدد العناصر. يجب تحديد أقصى عدد للعناصر في الجـــدول في متغير max_rec بحيث يمكن حجز مساحة التخزين الصحيحة واستخدامها. يجب اســـتخدام كلمة الأساس RESULTSET للإشارة إلى برنامج تشغيل ODBC بأنك تريده أن يقـــوم بتحويـــل البيانات التي يسترجعها إلى مجموعة سجلات (ينتج اصطلاح التسمية عن تنفيذ RDO).

ملاحظـــة: يجب أن يكون اسم الجدول الذي تستخدمه في سلسلة الأوامر نفس الاسم المحدد في الإجراء المحزم.

الشكل الثاني للصيغة أكثر طولاً، ولكنه يسمح لك بسحب كل جدول PL/SQL إلى الجزء الخاص به في مجموعة السجلات. (هذه هي الطريقة التي ستمكنك من الوصول إلى مجموعات سجلات متعددة من إجراء مخزن واحد في أوراكل). أن يجب هذه التقنية هي أن كل مجموعة سجلات سيكون لديها حقلاً واحداً فقط. على أي حال، تعتبر هذه النتيجة هي الغرض من آلية الجدول وطريقة وحدات أوراكل لـ OLE لمعالجته.

هناك ملاحظة أخيرة قبل الانتقال إلى استخدام التقنيات. بالإضافة إلى الصيغة التي تراها أعلاه، يمكننا أيضاً نواب لمواقع معاملات ("؟") في سلسلة الاستدعاء مثلما نفعل مع استدعاءات الإجراءات الأخرى. يعتبر ذلك أمراً هاماً لأن كثيراً ما سيكون الإجراء المخزن بحاجة إلى أن تتخطى أقصى عدد من السجلات التي تريد استرجاعها، الغرض الوحيد من وجود متغير max_recs في استدعاء الإجراء هو إخبار ODBC بما يحدث، ولن تصل القيمة إلى PL/SQL.

سنقوم الآن بوضع هذه التفاصيل موضع التنفيذ العملي عندما نقوم باسترجاع جدول PL/SQL كما رأينا في الفصل العاشر (انظر التعليمات البرمجية رقم ١٧-١٠). يجب أن يماثل تصميم سلسلة الاستدعاء تصميم الإجراء نفسه تماماً في حالة إجراء get_stars في حزمة movie_package، لدينا العنوان التالي:

```
PROCEDURE get_stars (
 p_table_size        NUMBER,
  p_stars        OUT char_table_type)
```

نحن نحتاج إلى تكوين معامل أمر واحد لإدخال أقصى حجم للجدول. يتم توضيح إعداد وحدة Command الموجودة في هذا المثال في التعليمات البرمجية رقم ١٧-٢١. توضح هذه التعليمات البرمجية كيفية إعداد وحدة Command بمعامل مدخلات وجدول مخرجات واحد. لاحظ أننا في هذه الحالة قمنا بإعداد المر بحيث يستطيع تلقي سجلاً فقط بجد أقصى. على أي حال، يحتاج الإجراء المخزن لمعرفة عدد السجلات التي سيقوم باسترجاعها فعلياً، ويستخدم معامل المدخلات لهذا الغرض. إذا كانت قيمة معامل المدخلات أكبر من ٢٠، سنتلقى رسالة التشغيل. يجب أيضاً أن تكون على دراية بأنه لا يستطيع أي من هذه الأرقام ضمان أنك ستتلقى عدد السجلات التي تم إرجاعها. قد يكون هناك سجلات غير كافية تناسب هذا المعيار.

تعليمات برمجة ١٧-٢١

```
Private Sub TestPLSQLTable()
   Dim qy As New ADODB.Command
   Dim rs As ADODB.Recordset
   Dim strSQL As String

'  First, set up the command object
   strSQL = "{call movie_package.get_stars " & _
         "(?, {RESULTSET 20, p_stars})}"
   With qy
      .CommandText = strSQL
      .CommandType = adCmdText
      .ActiveConnection = cn
      .Parameters.Append .CreateParameter(, adNumeric,
adParamInput)
   End With
'  Define the maximum number of records that we want back
'  and execute the stored procedure to return the Recordset.
   qy(0) = 5

   Set rs = New ADODB.Recordset
'   rs.CursorType = adOpenStatic
'   rs.LockType = adLockReadOnly
   Set rs = qy.Execute
   While Not rs.EOF
     MsgBox "Star name is  " & rs(0)
     rs.MoveNext
   Wend
   rs.Close
   Set rs = Nothing
End Sub
```

التجميع كلية

في هذا القسم، سنطلع على رموز تطبيق الأفلام الذي عرفناه في الفصل الرابع عشر. ســيتكون العديد من إجراءات هذا الإصدار من التطبيق مثل الفصل الرابع عشر تماماً، بالرغم من توفــر بعض هذه الإجراءات في هذا الفصل. تظهر الاختلافات الرئيسية في إجــراء CreateQueries الذي يتم استدعاءه في حدث Form_Load "انظــر التعليمــات البرمجيــة رقــم ١٧-٢٢" وإجــراء cmdAction الذي يتفاعل مع المســتخدم الــذي يقــوم بــالنقر علــى أزرار New وUpdate وDelete.

تعليمات برمجة ١٧-٢٢

```
Private Sub CreateQueries()
' Set up any queries required in this module.
    Dim strSQL As String
' Set up the query to call the insert stored procedure.
    With qyMovieInsert
        .CommandText = "video_store.movies_insert3"
        .CommandType = adCmdStoredProc
        .ActiveConnection = cn
        .Parameters.Append .CreateParameter _
           (, adNumeric, adParamInputOutput)
        .Parameters.Append .CreateParameter _
           (, adVarChar, adParamInput, 50)
        .Parameters.Append .CreateParameter _
           (, adNumeric, adParamInput)
        .Parameters.Append .CreateParameter _
           (, adVarChar, adParamInput, 6)
        .Parameters.Append .CreateParameter _
           (, adNumeric, adParamInput)
    End With
' Set up the query to select the participants.
    StrSQL = "     SELECT first_name || ' ' || last_name full_name,"
    StrSQL = strSQL & "   DECODE (role, 'A', 'Actor', 'D', 'Director')"
```

```
    StrSQL = strSQL & " FROM video_store.participants, "
    StrSQL = strSQL & "   video_store.movie_participants"
    StrSQL = strSQL & " WHERE movie_id = ? "
    StrSQL = strSQL & "  AND movie_participants.participant_id = " & _
              "    participants.participant_id "
    strSQL = strSQL & "  ORDER BY role"
    With qyParticipants
      .CommandText = strSQL
      .CommandType = adCmdText
      .ActiveConnection = cn
      .Parameters.Append .CreateParameter _
        (, adNumeric, adParamInput)
    End With
End Sub
```

في التعليمات البرمجية رقم ١٧-٢٢، نقوم بتكوين وحدتي أمر. الأولى هي qyMovieInsert التي رأيناها في بداية هذا الفصل. أي في نظام إنتاج، تقوم بتكوين وحدة Command مماثلة لكل عملية استدعاء إجراء مخزن تقوم بتنفيذها. يمكن أن تكون هذه مجموعة أساسية من الرموز ولكنها تجعل عملية استدعاء الإجراء المخزن أكثر سهولة وحدة Command الثانية هي qyParticipants وهي عبارة عن جملة SELECT مركبة نوعاً ما وتحتوي على معاملات. يتم تنفيذ وحدة Command هذه في كل مرة يتغير سجل الأفلام الحالي، لذلك يمكنك عرض المشاركين المناسبين في الفيلم. يعتبر استخدام استعلام ذو معاملات أكثر الطرق فعالية لإجراء ذلك.

توضح التعليمات البرمجية رقم ١٧-٢٣ إجراء cmdAction_Click الذي يتفاعل مع طلبات وظائف الإدراج والحذف. بالنسبة لهذا المثال، لدينا طريقتين لإجراء ذلك. يتم تنفيذ وظيفة الإدراج بواسطة وحدة Command بالرغم من أن وظيفة التحديث يتم تفيدها بواسطة سلسلة SQL ذات ترميز دائم. في الواقع، تقوم بإعداد استدعاءات فيجوال بيسك لكلتا هاتين العمليتين (غالباً في وحدة نمطية من طبقة الأفلام). بحيث تقوم بنقل هذا التعقيد خارج تعليمات البرمجة الطرف الأمامي للوسيط الرسومي الرئيسي، مثلما فعلنا مع إجراء movie_insert في مقدمة هذا الفصل.

تعليمات برمجة ١٧-٢٣

```
Private Sub cmdAction_Click(Index As Integer)
'  Execute the require action - insert, update or delete.
   Dim iField As Integer
   Dim strSQL As String

   If Index = 0 Then          ' Add a new record
      ShowNewRecord
      strMode = "Add"
ElseIf Index = 1 Then          ' Update the record
      If strMode = "Add" Then
         qyMovieInsert(0) = 0
         qyMovieInsert(1) = txtMovie(0)
         qyMovieInsert(2) = CInt(txtMovie(1))
         qyMovieInsert(3) = txtMovie(2)
         qyMovieInsert(4) = CInt(txtMovie(3))
         qyMovieInsert.Execute
         RefreshRecordset
Else
      '  Use a conventional inline call here
         strSQL = "BEGIN video_store.movies_update ("
         strSQL = strSQL & " rsMovies("movie_id") & ","
         strSQL = strSQL & "'" & txtMovie(0) & "', " ' Movie Name
         strSQL = strSQL & txtMovie(1) & ", "       ' Length
         strSQL = strSQL & "'" & txtMovie(2) & "', " ' Rating
         strSQL = strSQL & txtMovie(3)             ' Year
         strSQL = strSQL & "); END;"
         cn.Execute strSQL
      End If
      strMode = ""
      RefreshRecordset

   ElseIf Index = 2 Then          ' Delete the record
```

```
    cn.Execute "BEGIN video_store.movies_delete (" & _
          " rsMovies("movie_id") & "); END;"
    RefreshRecordset
  End If
End Sub
```

العملية غير المتزامنة

مع ADO 2.0، لدينا أخيراً نموذج تحت سيطرة الحدث يستطيع مناقشة، أو قد يفوق، RDO. لقد اطلعنا على الأحداث التي تعرضها ADO في الجداول ١٧-٣ و ١٧-٨ لوحدة Connection ووحدة Recordset على التوالي. يمكنك تحديد خيار عدم التزامن للأساليب التالية:

OpenOptions argument of the Connection.Open method
Options argument of the Recordset.Open method

يجب الإشارة للفصل الخامس عشر للإطلاع على شرحاً وافراً عن استخدام أحداث وعمليات غير متزامنة، تعمل تقنيات RDO بطريقة مشابهة لتقنيات ADO.

التعرف على هيكل قاعدة البيانات

كما فعلنا في الفصول القليلة السابقة، يعتبر فهم هيكل قاعدة البيانات التي نتعامل معها أمراً مفيداً في بعض الأحيان. مع ADO، لا يوجد لـ Tabledef الذي نراه في ADO أو جدول rdo الخاص بـ RDO. لدينا مجموعة من الحقول تحت وحدة Recordset. تحتوي ADO على سمات عديدة عن حقول مجموعة سجلات في خصائص وحدة Field (انظر الجدول ١٧-١٤).

الجدول ١٧-١٤: خصائص معامل الأمر

الخاصية	البيان
ActualSize	تشير هذه الخاصية إلى الحجم الفعلي للحقل.
Attributes	تحدد هذه الخاصية ما إذا كان المعامل يحتوي على أرقام موقعة وقيم خالية وبيانات RAW.

الجدول ١٧-١٤: خصائص معامل الأمر

الخاصية	البيان
DefinedSize	تشير هذه الخاصية إلى حجم الحقل كما هو محدد ف قاعدة البيانات قد يختلف ذلك عن قيمة الحجم الحقيقي، على سبيل المثال، لأعمدة VARCHAR2.
Name	تحتوي هذه الخاصية على اسم عشوائي للمعامل (يمكن أن يكون سلسلة طولها صفر).
NumericScale	توضح هذه الخاصية القيمة الأصلية للحقل عند ملء مجموعة السجلات.
OriginalValue	توضح هذه الخاصية القيمة الأصلبة للحقل عند ملء مجموعة السجلات.
Precision	توضح هذه الخاصية أقصى عدد من الأرقام في معامل عددي.
Type	توضح هذه الخاصية نوع بيانات المعامل الذي رأيناه في الجدول ١٧-١٧.
UnderlyingValue	توضح هذه الخاصية القيمة المثبتة الحالية في قاعدة البيانات.
Value	توضح هذه الخاصية القيمة الحالية للحقل في مجموعة السجلات.

إذا أردت فحص أنواع البيانات التي تحتوي عليها وحدة Field، يمكننا استخدام إجراء مثل ذلك الذي يوجد في التعليمات البرمجية رقم ١٧-٢٤.

تعليمات برمجة ١٧-٢٤

```
Private Function FieldType(intType As Integer) As String
' Convert the ADO field type to an Oracle column type
 Select Case intType
   Case adInteger
     FieldType = "NUMBER"
```

```
        Case adDecimal
          FieldType = "NUMBER"
        Case adNumeric
          FieldType = "NUMBER"
        Case adSingle
          FieldType = "NUMBER"
        Case adDouble
          FieldType = "NUMBER"
        Case adDate
          FieldType = "DATE"
        Case adChar
          FieldType = "CHAR2"
        Case adVarChar
          FieldType = "VARCHAR2"
        Case adLongVarChar
          FieldType = "LONG"
        Case adLongVarBinary
          FieldType = "LONG RAW"
        Case adVarBinary
          FieldType = "RAW"
        Case adDBTimeStamp
          FieldType = "Date"
        Case Else
          FieldType = "Unknown " & intType
    End Select
End Function
```

أسلوب OpenSchema

كما عرفت من خلال المعلومات المتاحة للخقول، كما هو موضح أعـــلاه، لا تقـدم لــك ADO بصورة افتراضية معلومات وافرة عن وحدات قاعدة البيانات التي يمكنك الوصول إليها. يعتــبر ذلك قرار تصميم جيد بالمقارنة مع DAO/JET التي تجلب ما تستطيع عن وحدات قاعدة البيانات عندما تقوم في البداية بالاتصال مع قاعدة البيانات. في اغلب الأوقات لا يتم مطلقاً استخدام هــذه المعلومات التي تأخذ وقتاً طويلاً وموارد هامة في تجميعها، وسيحدث ذلك لكل اتصال يتم تنفيذه.

هناك بديل لـــ ADO شديد الفعالية، يسمح لك باستخدام أسلوب OpenSchema في مقـــابل الاتصال إحضار كل معلومة تخص وحدات قاعدة البيانات. يعتبر اسم OpenSchema نوعاً مـــا من وجهة نظر أوراكل، لأنه لا يقوم باسترجاع كل وحدة مخطط قاعدة البيانات تستطيع الوصول إليه.

صيغة أسلوب OpenSchema

```
Set rs = cn.OpenSchema (QueryType, Criteria, SchemaID)
```

المتغيرات الخاصة بأسلوب OpenSchema

QueryType: يوضح هذا المتغير نوع المعلومات التي تريـــد اســـترجاعها. هنـــاك ثلاثون نوعاً، يتم توضيح بعض المعلومات المفيدة في الجدول ١٧-١٥.

Criteria: يمكنك استخدام هذا المتغير لتحديد مجموعة فرعية من كافة السمات التـــي يمكن إرجاعها بشأن وحدة مخطط قاعدة البيانات. يعتبر هذا المتغير اختياريا. إذا قمـــت بحذفه، ستقوم باسترجاع كافة السمات في مجموعة السجلات.

الجدول ١٧-١٥: QueryTypes المتاحة لطريقة OpenSchema.

نوع الاستعلام	البيـان
AdSchemaColumns	معلومات الجدول والعمود (يستخدم هذا الخيار بعناية لأنه مـــن الممكن الحصول على الكثير من المعلومات)
AdSchemaForeignKeys	مفاتيح غريبة
AdSchemaIndexes	فهارس
AdSchemaPrimaryKeys	مفاتيح أولية
AdSchemaProcedureParameters	اسم الإجراء وأنواع المعامل
AdSchemaTables	كل الجداول التي لك حق قراءتها

سوف تجد دائماً عند استخدام هذه الأنواع من الاستعلامات حقلاً يصف "Catalog" وتلك هي عادة SQL Server الخاص بمايكروسوفت غير أن ذلك لا ينطبق على أوراكل، ومن أحد المميزات هي أنه من الممكن استعادة الإجراءات المخزنة الخاصة بأوراكل التي توجد داخل الحزم البرمجية بالإضافة إلى الوظائف، إلا أنه من غير الممكن استعادة عنوان الحزمة البرمجية.

تحذير عند استخدام طريقة OpenSchema سوف تجد أن المجموعة الفرعية من QueryTypes هي فقط التي تعمل مع مشغل ODBC الخاص بمايكروسوفت، ومن المستحسن استخدام OLE-DB.

معالجة الأخطاء

عند استخدام أسلوب الوصول للبيانات سواء كان ADO أو أسلوب آخر، تتوفر لديك إذن إمكانية (قد يقول البعض احتمالية) وجود رسائل أخطاء نظراً لتفاعلك مع مصدر البيانات فضلاً عن أخطاء برمجة فيجوال بيسك المعتادة. مثلما توفر الأساليب الأخرى مجموعة إضافية لحمل أخطاء الوصول للبيانات هذه فإن ADO أيضاً لديها مجموعة الأخطاء. تتكون هذه المجموعة من وحدة خطأ واحدة أو أكثر، تتضمن كل واحدة منها على الخصائص الموضحة في الجدول ١٧-١٦. لأن المجموعة تحتوي على أخطاء يتم مواجهتها على مصدر البيانات، تكون المجموعة تابعة لوحدة Connction الخاصة بمصدر البيانات.

الجدول ١٧-١٦: خصائص وحدة الخطأ الخاصة بـ ADO

الخاصية	البيان
Description	توفر هذه الخاصية بياناً للخطأ.
HelpContext	توفر هذه الخاصية معرف سياق ملف التعليمات لرقم الخطأ.
HelpFile	توفر هذه الخاصية الاسم الكامل لملف تعليمات فيجوال بيسك.
NativeError	توفر هذه الخاصية رقم الخطأ من مصدر البيانات.
Number	توفر هذه الخاصية رقم خطأ فريد. هذه الخاصية قليلة الاستخدام.
Source	توفر هذه الخاصية اسم وحدة ADO التي إصدارات الخطأ.

الجدول ١٧-١٦: خصائص وحدة الخطأ الخاصة بـ ADO

الخاصية	البيان
SQLState	توفر هذه الخاصية سلسلة من خمسة حروف لمعايير ANSI تحتوي على تعليمات البرمجة الخطأ من مزود OLE-DE.

يمكنك التحلق في مجموعة الأخطاء بواسطة تعليمات البرمجة التالية:

```
Dim Error As ADODB.Error
For Each Error In cn.Errors
   Debug.Print Error.Description & Chr(9) & Error.NativeError
Next
```

عادة ستجد أنها أول وحدة خطأ في المجموعة التي تستحق، هي فقط، الإطلاع عليها. على سبيل المثال، تعليمات البرمجة التالية:

```
Debug.Print "Description " & cn.Errors(0).Description
Debug.Print "NativeError " & cn.Errors(0).NativeError
Debug.Print "Number      " & cn.Errors(0).Number
Debug.Print "Source      " & cn.Errors(0).Source
Debug.Print "SQLState    " & cn.Errors(0).SQLState
```

يمكن أن ينتج المخرجات الموضحة أدناه:

```
Description [Microsoft][ODBC driver for Oracle][Oracle]ORA-00904:
invalid column name
NativeError 904
Number      -2147217900
Source      Microsoft OLE DB Provider for ODBC Drivers
SQLState    S0022
```

كالمعتاد، البند الرئيسي الهام هو البيان. يجب أن يتم تحليل ذلك لاستخلاص المعلومات الأكثر أهمية وهي في هذه الحالات، invalid column name إن ADO لديها خاصية NativeError التي توضح رقم خطأ أوراكل الحقيقي "وهو ٩٠٤ هنا". في ADO توجد هذه القيمة في Errors(0).Number.

تستطيع ADO إرجاع رسائل الخطأ التي يمكنك التقاطها بصورة طبيعي. هذه الأخطاء هي تلك التي لم يصدرها مصدر البيانات. بالإضافة إلى ذلك، ستجد أن الأحداث تقوم بإرجاع رسائل خطأ.

استخدام بيئة البيانات

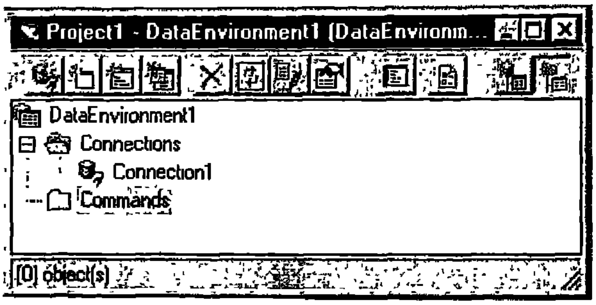
عند كتابة تعليمات البرمجة لكي تتعامل مع استدعاءات الإجراءات المخزنة لفترة من الوقت، ستبدأ في التفكير في أن لابد من وجود طريقة أفضل هي كتابة هذه المعلومات الكثيرة في كل مرة تريد تشغيل الإجراء. إنك تحتاج إلى طريقة آلية بصورة أكبر. يمكنك تحقيق هذا الهدف، وهو تقليل كتابة الرموز، عن طريق معالج البيانات التي تقوم بتكوين نموذج بعناصر تحكم ملتزمة بالبيانات ترتبط بمجموعة سجلات محددة، ولكن هذه الطريق عبر شاملة أو قابلة للتدرج.

في الإصدارات السابقة لفجيوال بيسك، تستطيع RDO استخدام وحدة UserConnection لتسهيل استخدام الإجراءات المخزنة والاستعلامات تم إصدار فجوال بيسك ٦ بيئة البيانات "Data Environment"، التي هي عبارة عند مصمم ActiveX كبير ومعقد يهدف إلى تبسيط مهام المطورون. في هذا القسم، فسيلقي نظرة سريعة على Data Environment لتسهيل استدعاء الإجراءات المخزنة. يستطيع القيام بمزيد من الأشياء ولكنك هنا يجب أن تكون حذراً نظراً لتعقيده ومشاكل توازنه التي يمكن أن تسبب كثير من المشاكل.

عن طريق مصمم Data Environment يمكنك تكوين وحدة Data Environment تعالج كل الاتصالات بقاعدة البيانات (أو القدر الذي تريده منها). على سبيل المثال، يمكنك تصميم إمكانية استدعاءات الإجراءات المخزنة في Data Environment. يمكنك تكوين مجموعات سجلات وإسقاط الحقول أو مجموعة السجلات بأكملها على نموذج، ويكون لديك نموذج لحظي لصيانة البيانات. على أي حال لديك عملاً كثيراً لتشغيل التطبيق (مثل استدعاء إجراءات مخزنة)، إلا إذا كنت تستخدم مجموعات سجلات قابلة للتحديث.

سنقوم بتكوين Data Environment جديدة وإعدادها تنفيذ إجراء مخزن. في حالة استخدام هذه الطريقة في موقف عملي، قد تقوم بإعداد وحدة Data Environment لكل مساحة قاعدة بيانات في المثال، ستقوم بإعداد بيئة لكل قاعدة بيانات Movie ولأن Data Environment تختلف قليلا عن عمليات فجوال بيسك العادية، سنتناولها تقليل من التمهل.

ستبدأ بتكوين مشروع جديد نسميه DeDemo. إذا قمـــت بنقـر Project ⇦ Add Data Environment، سيتم إضافة وحدة جديدة للمشروع "انظر الشكل ١٧-٦". سيكون اسم الوحـــدة والملف الذي يتم تخزينها فيه Data Environment بصورة افتراضية، ولكننا سنتأكد من تغيـير كلا هذين الاسمين. إذا نظرت على نافذة المشروع، ستجد أيضاً أن المصمـم قـد تـم إضافتـه للمشروع.



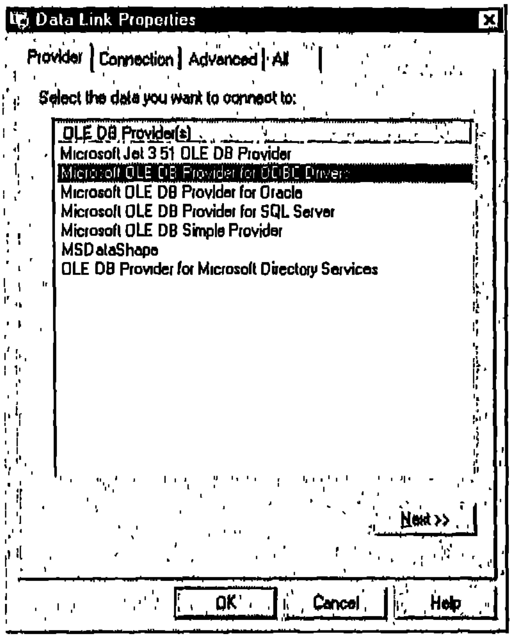
الشكل ١٧-٦: عنصـــر Data Environment الجديد.

أول شيء أريد القيام به هو إعادة تسمية وحدة DE (Data Environment) بحيث لا يكـون من الواجب تعديل أي عمل تالي في حالة قيامنا بتغيير الاسم. لإجراء ذلك، انقر مرة واحدة علـى نافذة DE واضغط زر F$ لإظهار الخصائص، ثم قم بتغيير الاسم (في هذه الحالة، أطلق عليـها اسم Movies). الآن يعتبر حفظ DE في ملف فكرة حساسة. يمكنك عمل ذلك لحفظ الملف. كمـا هو الحال مع كافة حلقات المصمم، سيحصل هذا الملف علـى الامتـداد dsr. إذا كنـت تنـوي استخدام DEs بصورة أكبر، يجب إعداد مجلد خاص لها بحيث يمكن الاشتراك فيها بسهولة.

في الشكل ١٧-٦، يجب أن تكون قد لاحظت أن لدينا نوعين من المجلات فـي DE، همـا Connections أو "اتصـالات" Commands "أوامر". يوجـــد وحـدة Connection فـي DE بصورة افتراضية، لأنها إن لم تكن موجودة، لن يمكنك عمل أي شيء، لكن وحدة Connection هذه تحتاج لأن يتم إعدادها بالخصائص المناسبة.

انقر بالزر الأيمن على Connection 1 "يمكنك إعادة تسميعها إذا أردت ذلـك"، ثـم اخـتر Properties من القائمة المنبثقة. ستصبح النافذة التي تراها الآن مشـــابهة لتلـك التـي قمـت باستخدامها لتكوين وصلة بيانات "Data Link" (انظر الشكل رقم ١٧-٧). حدد مـزود OLE-DE وقم بإدخال معلومات الاتصال (اسم المستخدم وكلمة السر وإما اسم الخدمـة الخـاص بـالمزود

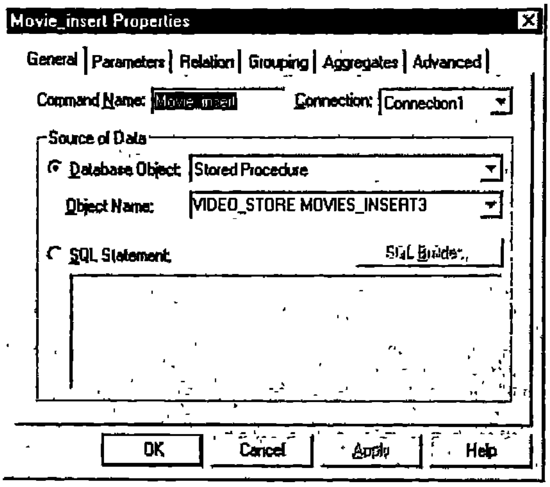
الأصلي أو Dsn الخاص بمصدر البيانات ODBC). والآن لدينا اتصال بيانات صحيح، فنحن لسنا بحاجة لتحديد سلسلة الوصل في أي مكان بتعليمات البرمجة، يعد ذلك من ضمن الأشياء المختفية في DE.



الشكل ١٧-٧ تكوين اتصال البيانات.

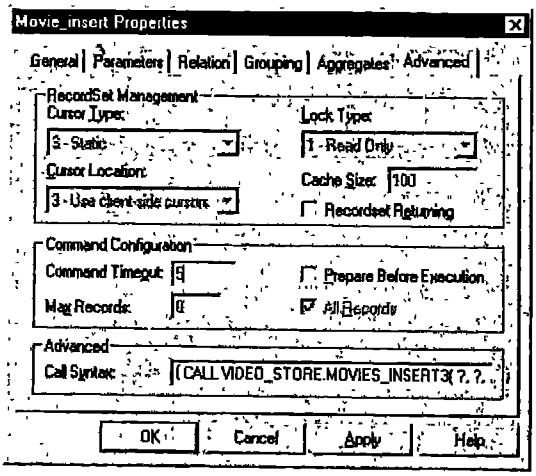
تحذير يجب أن تتذكر أن كل DE لديها الاتصال الخاص بها. إذا قمت باستخدام أكثر من DE واحدة أو قمت بتكوين اتصالاً آخر باستخدام الطريقة التقليدية التي نستخدمها حتى الآن، فسوف تنتهي باتصالات متعددة لنفس قاعدة البيانات.

لتتمكن من عمل أي شيء بـ DE، نحتاج لإضافة وحدة Command واحدة على الأقل. يمكنك إجراء ذلك عن طريق النقر بالزر الأيمن على أي وحدة في نافذة DE واختيار "Add Command" من القائمة المنبثقة. سيظهر أمر جديد في نافذة DE تحت مجلد Commands. يساوي ذلك وحدة Command منفردة، وإلى أن تقوم بتعيين خصائصها، لا يمكنها عمل شيء. انقر بالزر الأيمن على وحدة Command واختر "...Properties" من القائمة المنبثقة. انظر الشكل رقم ١٧-٨.



الشكل ١٧-٨ خصائص وحدة Command.

الآن يجب أن نقوم بتسمية وحدة Command، يظهر المثال movie_insert. يجب أن تقوم بإعداد الاتصال لهذا الأمر، وهو ذلك الذي قمنا بتكوينه في بداية هذا المثال (Connection1). بعد ذلك، يجب أن تحدد نوع المر الذي تريد تكوينه. أمامنا اختيار بين تنفيذ إجراء مخزن (أو وظيفة). وتكوين مجموعة سجلات من جملة SQL. في المثال، نقوم باستخدام الإجراء المخزن movies_insert3 مع المثال الذي قمنا باستخدامه في هذا الفصل والفصول الأخرى. عند إلقاء نظرة سريعة على علامة التبويب Advanced، سيتضح لك أن DE قد استجوبت قاعدة البيانات وأنها تعرف أن لا تقوم بإرجاع مجموعة سجلات وأنها قامت بتكوين استدعاء الإجراء المخزن بنواب مواقع معاملات (انظر الشكل ١٧-٩). إذا كنا نقوم بتكوين أمر جملة SQL،إذن سيكون لدينا بعض السيطرة على مجموعة السجلات الناتجة على الصفحة من مربع حوار الخصائص.



الشكل ١٧-٩ علامة التبويب Advanced على مصمم خصائص الأمر.

عندما تقوم أخيراً بالنقر على OK، فسنكون قد قمنا بإعداد DE بالمر الخاص بــــها. الشـــيء الثاني هو معرفة كيفية دمجها في المشروع. وحدة DE هي عبارة عن عنصـــر تحكـم Achiex ويجب أن تقوم بتكوين مثالاً لها بالرموز لكي نستطيع استخدامها. نستطيع إجراء ذلــك بواســطة سطر الرموز التالي:

```
Private DE As New Movies
```

يعلن ذلك DE جديدة كمثال لــ DE التي قمنـــا بإعدادهـــا وســميتها Movies والآن، نريـــد الإطلاع على الرموز الخاصة لتنفيذها (انظر التعليمات البرمجية رقم ١٧-٢٥). لاحظ أن الجزء العامل في هذه الرموز هي سطر واحد وأننا لا نحتاج لإعداد أي استعلام على الإطلاق. لا يمكنك تسهيل استدعاء إجراء مخزن. في الاستدعاء، تستطيع إتاحة قيم فعلية لكل معاملات المخرجــات، ولكن بالنسبة لمعامل المدخلات/المخرجات (الأول)، تحتاج لتحريره كمتغير بحيث يكون لـــــ] شيئاً تقوم بإرجاع القيمة له.

تعليمات برمجة ١٧-٢٥

```
Private Sub MovieInsert(iMovieId As Integer, strMovieName As String,
_
        iLength As Integer, strRating As String, iYear As Integer)

   DE.Movie_insert iMovieId, strMovieName, iLength, strRating, iYear

   MsgBox "The movie id used was " & iMovieId
End Sub
```

تحـــذيـــر هناك مشاكل شديدة مصاحبة لــ DE، يشير إلـــى إنـــها لـــم تصبـــح بعـــد منتجا كاملا، كما يوضح عدم نضجها. بالنســـبة لأوراكـــل، لا تقـــوم DE بدعم الإجراءات المحزمـــة، ويجـــب حقيقـــة أن تحـــاول اســـتخدام هـــذه الإجراءات المحزمة كلما تستطيع ذلك ليـــس فقـــط مـــن وجهـــة النظـــر التي تتعلق بالصيانة ولكن أيضا لأنها أكثر فعاليـــة. مـــن ناحيـــة أخـــرى، بالنسبة لإرجـــاع مجموعـــات الســـجلات، لا تـــزال البيئـــة لديـــها مـــن المشكلات. النصيحة المثلى هي أن تقوم بالاختيـــار عندمـــا يكـــون لديـــك الكثير من وقت الفراغ ولا تعتمد على DE فـــي عملـــك.

عيوب ADO

تعتبر ADO حالياً ذروة تقنيات مايكروسوفت. كما تعتبر ADO2.0 أكثر نضجاً من إصـدار ١,٥ خاصاً مع نماذج الأحداث. ولكن لا يزال هناك أشياء كثيرة مفقودة من هذه التقنية.

- لا يمكنك استرجاع متغيرات مؤشر PL/SQL من الإجراءات المخزنة.
- لا يمكنك استخدام معاملات الربط الخاصة بأوراكل.
- تعتبر Date Environment غير مكتملة النضج نسبياً بسبب ما بها من مشكلات.
- لا تدعم Date Environment إجراءات المحزمة لأوراكل.

خلاصة

يتناول هذا الفصل وحدات بيانات ActiveX من وجهة نظر مستخدم أوراكل. لقد اطلعنـا علـى نموذج ADO الذي يعتبر أحد نماذج تقنيات الوصول للبيانات الموضحة في هذا الكتـاب. كمـا اطلعنا على كيفية تكوين اتصال بقاعدة بيانات أوراكل باستخدام كل من مزود OL-DB الخـاص بـ ODBC ومزود OLE-DB الأصلي لأوراكل. وقمنا باستخدام برنامج تشغيل ODBC لأنه أكثر قوة وثباتا شيئا ما من المزود الأصلي.

انتقلنا بعد ذلك إلى الطرق المختلفة لبناء مجموعات سجلات اطلعنا على معالجتها على كيفيـة معالجتها. أيضا، قمنا بتغطية شاملة لاستخدام الأحداث التي توفرها لنا الاتصالات ومجموعـات السجلات. بعد مجموعات السجلات، قمنا بالتعامل مع وحدة Command. بالإضافة إلى كيفيـة إنتاج استعلامات ذات معاملات واستدعاء إجراءات مخزنـة ووظـائف بمعـاملات مدخـلات ومخرجات. اطلعنا على كيفية استرجاع جداول PL/SQL من إجراءات مخزنة إلى مجموعـات سجلات، ثم تناولت بالتفصيل معالجة الأخطاء الخاصة بـ ADO ويكل قاعدة البيانات. وأخـيرا، انتهينا بإلقاء نظرة على كيفية استخدام بيئة البيانات (DE) لتنفيذ إجراءات مخزنة بأقل قدر مـن الرموز الخاصة بفيجوال بيسك.

١٨

وحدات أوراكل التي تخص OLE

كل التقنيات الخاصة بفيجوال بيسك التي تم عرضها حتى الآن هي تقنيات تقدمها مايكروسوفت.غير أنه هناك عدد غير قليل من التقنيات الأخرى مثل وحدات OLE الخاصة بأوراكل (OO4O) وهي أكثرها شهرة. ويعد OO4O هو الاستجابة الفعلية لبرنامج DAO الذي تم إنشاؤه عليه.غير أنه قد تغير عدة مرات منذ الإصدار الأول له الخاص بالفيجوال بيسك ٣. وكما ذكرنا في الفصول السابقة، فإنه لا توجد طريقة للوصول إلى البيانات من نوع ODBC تسمح بالفعل للمبرمج باستخدام كل الميزات الخاصة بأوراكل. وقد تم تعزيز OO4O بعدد من المميزات في السنوات الأخيرة والذي صاحبه تعزيز برنامج أوراكل ببعض الإمكانات مثل استعادة جداول PL/SQL ومتغيرات المؤشر من الإجراءات المخزنة. وبوجه عام فإن هذه المميزات لا تتوافق مع أوراكل غير أنه من المحتمل أن يحتوي OLE-DB على هذه الإمكانية في أحد الأيام. ومن أحد المميزات التي يتمتع بها هذا المنتج هي أنه يعد الطريقة الوحيدة للوصول إلى البيانات التي تعالج الامتداد المتعلق بأوراكل على الرغم من أن هذه الإمكانية ما زالت غير متاحة الآن.

وتعد طريقة OO4O في غاية الأمن عند تطبيقها على Microsoft Transaction Server (MTS) وInternet Information Server (IIS). وهي عبارة عن طريقة للوصول إلى البيانات تتكون فقط من ٣٢ بت. كما أنها مختلفة عن الطرق الأخرى للوصول إلى البيانات في أنها الطريقة الوحيدة التي تستخدم خادم OLE جاري التشغيل. وهو نفس النوع من الخادم الجاري التشغيل الذي تستطيع أن تقوم بتكوينه باعتباره dll في بيئة فيجوال بيسك رغم أنه في تلك الحالة يتم تطويره في ++C. كما يستخدم COM لنقل البيانات بين التطبيق وOO4O، لذلك يجب توخي الحذر عند معالجة البيانات.

ويقوم OO4O بعرض التكامل مع أوراكل الذي يقوم RDO مع SQL Server. وفي الواقع فإن التكامل مع هذا التطبيق يعد تكاملاً وثيقاً حتى المرحلة التي يصبح فيها OO4O من أحد الإمتدادات الخاصة بخادم أوراكل.

وتتضمن الموضوعات التي يتم تناولها في هذا الفصل:

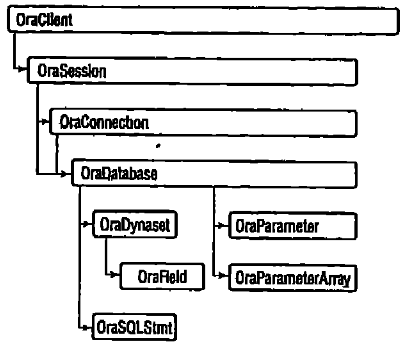
- التدرج الخاص بطريقة OO4O.
- موضوع الربط المبكر.
- الوصول إلى قاعدة البيانات.
- تكوين OraDynasets.
- استخدام المعاملات.
- تنفيذ أوامر SQL.
- استدعاء الإجراء المخزن.

- استعادة متغيرات المؤشر وجداول PL/SQL.
- استخدام OraSQLStmts.
- عرض واستخدام بنية قاعدة البيانات.
- عيوب OO40.

تدرج وحدات OLE الخاصة بأوراكل

كما هو الحال مع كل الطرق المستخدمة في التعامل بين الفيجوال بيسك وأوراكل، فإن طريقة OO40 تقوم بعرض نموذج الوحدة الخاص بأوراكل والذي من شأنه أن يسهل من عملية التعامل مع البيانات. ويتضح من تدرج الوحدات في الشكل ١٨-١ أنها مشتقة من DAO مع إضافة بعض الإمتدادات. وتتشابه التعبيرات مع DAO مما يسهل التعامل مع OO40 حتى بالنسبة للمستخدم حديث العهد به.ويتم تفصيل الوحدات الأولى في الترتيب في الجزء التالي، ثم يتم عرض الوحدات السفلي في الترتيب المسؤولة عن معظم الوظائف في بقية هذا الفصل.

الشكل ١٨-١
تدرج وحدات Data Access Objects



OraClient

تتضح بعض الاختلافات في وحدة OraClient عند الاستمرار في تتبع الجدول. وعلى الرغم من أنها العنصر الأول في الشكل إلا أنها تمثل بيئة العمل التي يتم فيها تشغيل OO40. ومن الممكن اعتبارها خادم جاري تشغيله. وعلى الرغم من أنها تحوي وحدات OraSession إلا أنها من الممكن أن يتم تجاهلها.

OraSession

على العكس من وحدة OraClient، فإن وحدة OraSession هي الوحدة العملية ذات أعلى مستوى في التدرج عند الأخذ في الإعتبار وجود التطبيق. ويمكن تكوين وحدات جلسات العمل

كل التقنيات الخاصة بفيجوال بيسك التي تم عرضها حتى الآن هي تقنيات تقدمها مايكروسوفت.غير أنه هناك عدد غير قليل من التقنيات الأخرى مثل وحدات OLE الخاصة بأوراكل (OO4O) وهي أكثرها شهرة. ويعد OO4O هو الاستجابة الفعلية لبرنامج DAO الذي تم إنشاؤه عليه.غير أنه قد تغير عدة مرات منذ الإصدار الأول له الخاص بالفيجوال بيسك ٣. وكما ذكرنا في الفصول السابقة، فإنه لا توجد طريقة للوصول إلى البيانات من نوع ODBC تسمح بالفعل للمبرمج باستخدام كل الميزات الخاصة بأوراكل. وقد تم تعزيز OO4O بعدد من المميزات في السنوات الأخيرة والذي صاحبه تعزيز برنامج أوراكل ببعض الإمكانات مثل استعادة جداول PL/SQL ومتغيرات المؤشر من الإجراءات المخزنة. وبوجه عام فإن هذه المميزات لا تتوافق مع أوراكل غير أنه من المحتمل أن يحتوي OLE-DB على هذه الإمكانية في أحد الأيام. ومن أحد المميزات التي يتمتع بها هذا المنتج هي أنه يعد الطريقة الوحيدة للوصول إلى البيانات التي تعالج الامتداد المتعلق بأوراكل على الرغم من أن هذه الإمكانية ما زالت غير متاحة الآن.

وتعد طريقة OO4O في غاية الأمن عند تطبيقها على Microsoft Transaction Server (MTS) وInternet Information Server (IIS). وهي عبارة عن طريقة للوصول إلى البيانات تتكون فقط من ٣٢ بت. كما أنها مختلفة عن الطرق الأخرى للوصول إلى البيانات في أنها الطريقة الوحيدة التي تستخدم خادم OLE جاري التشغيل. وهو نفس النوع من الخادم الجاري التشغيل الذي تستطيع أن تقوم بتكوينه باعتباره dll في بيئة فيجوال بيسك رغم أنه في تلك الحالة يتم تطويره في ++C. كما يستخدم COM لنقل البيانات بين التطبيق وOO4O، لذلك يجب توخي الحذر عند معالجة البيانات.

ويقوم OO4O بعرض التكامل مع أوراكل الذي يقوم RDO مع SQL Server. وفي الواقع فإن التكامل مع هذا التطبيق يعد تكاملاً وثيقاً حتى المرحلة التي يصبح فيها OO4O من أحد الإمتدادات الخاصة بخادم أوراكل.

وتتضمن الموضوعات التي يتم تناولها في هذا الفصل:

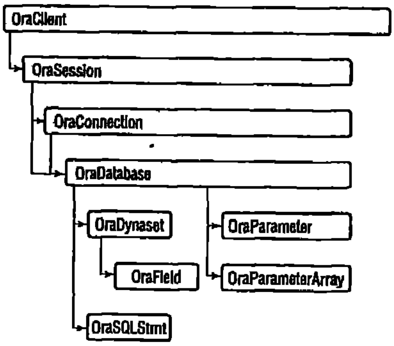
- التدرج الخاص بطريقة OO4O.
- موضوع الربط المبكر.
- الوصول إلى قاعدة البيانات.
- تكوين OraDynasets.
- استخدام المعاملات.
- تنفيذ أوامر SQL.
- استدعاء الإجراء المخزن.

- استعادة متغيرات المؤشـــر وجـداول PL/SQL.
- اســـتخدام OraSQLStmts.
- عرض واستخدام بنية قــاعدة البيانـــات.
- عيــوب OO40.

تدرج وحدات OLE الخاصة بأوراكل

كما هو الحال مع كل الطرق المستخدمة في التعامل بين الفيجوال بيسك وأوراكل، فـــإن طريقـــة OO40 تقوم بعرض نموذج الوحدة الخاص بأوراكل والذي من شأنه أن يسهل من عملية التعامل مع البيانات. ويتضح من تدرج الوحدات في الشكل ١٨-١ أنها مشتقة من DAO مع إضافة بعض الإمتدادات. وتتشابه التعبيرات مع DAO مما يسهل التعامل مع OO40 حتى بالنسبة للمســـتخدم حديث العهد به.ويتم تفصيل الوحدات الأولى في الترتيب في الجزء التـــالي، ثـم يتـم عـرض الوحدات السفلي في الترتيب المسؤولة عن معظم الوظائف في بقية هذا الفصل.

الشكل ١٨-١
تدرج وحدات Data Access Objects



OraClient

تتضح بعض الاختلافات في وحدة OraClient عند الاستمرار في تتبع الجدول. وعلى الرغـــم من أنها العنصر الأول في الشكل إلا أنها تمثل بيئة العمل التي يتم فيها تشـــغيل OO40. ومـــن الممكن اعتبارها خادم جاري تشغيله. وعلى الرغم من أنها تحوي وحدات OraSession إلا أنـها من الممكن أن يتم تجاهلها.

OraSession

على العكس من وحدة OraClient، فإن وحدة OraSession هي الوحـدة العمليـــة ذات أعلـــى مستوى في التدرج عند الأخذ في الإعتبار وجود التطبيق. ويمكن تكوين وحدات جلسات العمـــل

عن طريق استدعاء مباشر من الخادم الجاري تشغيله (عبر طريقة الأتمتة التي تخص CreateObject) باعتبارها معاكسة لطريقة استدعاء OO4O. ويقوم OraSession بتوفير وحدة مستوى المعاملة (ولذلك فهي تقوم بتدعيم BegiTrans وCommitTrans). وإذا كنت تحتاج إلى أكثر من مجال واحد للمعاملة، فإن ذلك يعني أنك تحتاج لتكوين OraSession أخرى. وهي تحتوي أيضاً على وحدات OraDatabase ووحدات OraConnection.

OraConnection

تعد وحدات OraConnection إعادة عرض تقديمي للاتصال بين وحدة OraDatabase وخادم أوراكل.ومن الناحية العملية فإنها تقوم بتدعيم طريقة CommitTrans وطريقة RollbackTrans والذي يتم التعامل معهم عل نحو أمثل باستخدام وحدات OraSession. ولن يتم التعرض إلى OraConnection ومن الممكن أن يتم تجاهلها.

OraDatabase

تمثل وحدة OraDatabase دورة عمل واحدة مع قاعدة البيانات. وهي تتشابه مع جلسات العمل لأوراكل في أي مجال آخر، مع اسم المستخدم وكلمة السر. وقد كان من الممكن في الإصدارات السابقة من OO4O أن يتم مشاركة جلسات العمل إذا كان اسم المستخدم هو كلمة السر، غير انه بداية من الإصدار ٢-٢ والإصدارات التالية له فإن OraDatabase لها وحدة Session خاصة بها. ومن الممكن أن تحتوي كل OraDatabase على واحدة أو أكثر من وحدات OraDynaset لاستعادة الصفوف من قاعدة البيانات.

موضوعات الربط المبكر

يقوم الفيجوال بيسك بتحديد المعاملات المطلوبة عندما يتم استدعاء أية خاصية أو طريقة في مكون ActiveX مثل OO4O. وتعتمد الطريقة المستخدمة للقيام بهذه المهمة على الطريقة التي تستخدمها لتعريف هذه الوحدة. وفي الإصدارات السابقة للإصدار ٢-٢ كان يتم تعريف وحدات OO4O على أساس أنها "Object". وهذا يعني أنها نوع عام من البيانات الذي يقوم فيجوال بيسك بمعرفة نوع وحدة OO4O عن طريق الإشارة إلى نوع المكتبة وأي المعاملات يتم استخدامه للاستدعاء في وقت التشغيل. وهذا يسمى الربط المتأخر وهو يتطلب مصروفات رأسية إضافية مقارنة بالتعريف عن العنصر بشكل ملائم.

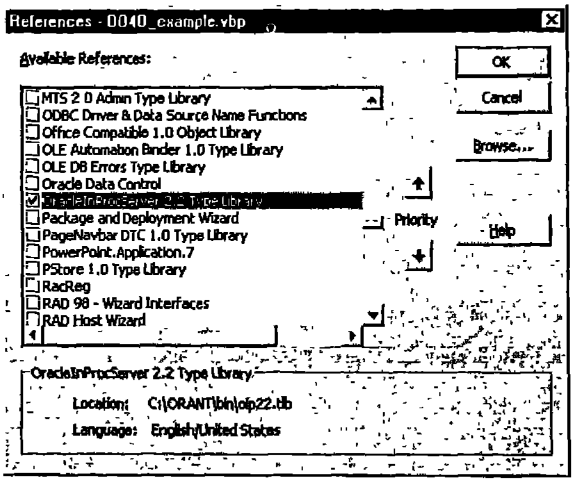
ومن الممكن تعريف الوحدات باستخدام النوع الفعلي لها باستخدام الإصدار ٢-٢.ويمكن هذا التعريف برنامج الفيجوال بيسك من البحث عن نوع الوحدة وبعض المعلومات عن المعامل في وقت التجميع كما تعمل على جعل العملية تتم بشكل أسرع في وقت التشغيل. وتسمى هذه الطريقة

بالربط المبكر كما أنها تتطلب أن تقوم بدمج أحد المراجع الجديدة إلى المشروع. ولإضافة هذا المرجع يجب اتباع الخطوات التالية:

١- اختر Project ⇦ References لفتح مربع حوار References كما هو موضح في الشكل ١٨-٢. وسوف تظهر قائمة بالوحدات التي يمكن أن تقوم بتضمينها في التطبيق.

٢- اختر الخيار OraclInProServer 2-2 Type Library.

٣- انقر OK.



الشكل ١٨-٢
مربع حوار Refrences مع Oracle In-Process Server Type Library الذي تم اختياره.

تلميح إذا كنت تقوم بتعريف الوحدات ووجدت أن ميزة Auto List Members لا تقوم بتحديد أية وحدات خاصة بأوراكل فإن هذا يعني أنك لم تقم بتضمين مرجع Oracle In-Process Server.

تحذير قد تواجه بعض الصعوبات في الربط المبكر إذا كنت تعمل باستخدام فيجوال بيسك ٦. عند حدوث ذلك فإنه يجب عليك أن تقوم بالتحويل إلى آخر ربط ثم قم بتعريف وحداتك على شكل "As Objects". وبسبب هذه المشكلة تستخدم كل الأمثلة الموجودة في هذا الفصل ترابط متأخر.

الوصول إلى قاعدة البيانات

يعد الوصول إلى قاعدة البيانات باستخدام OO4O أسهل طرق الدخول إلى قاعدة بيانات أوراكل مع فيجوال بيسك، رغم أنها قد تبدو معقدة للوهلة الأولى. ومن أحد المميزات هو أن التعليمات البرمجية لا تختلف تقريباً، كما أنك تحتاج فقط إلى قدر بسيط من المعلومات حتى يتسنى لك القيام بعملية الاتصال.ويجب استدعاء طريقة Visual Basic CreateObject.حتى تتمكن من تكوين وحدة OraSession جديدة كما تقوم بضبط المرجع إلى متغير الوحدة الذي تم تعريفه. ولن يطرأ على هذه التعليمات البرمجية أية تغييرات بغض النظر عن عدد الجلسات التي قد قمت بتكوينها قم فقط بتعيين متغيرات مختلفة للوحدة.

وما تحتاج عمله أولا هو تكوين OraSession، وهي تتم كالتالي:

```
Dim OraSession As OraSession
Set OraSession = CreateObject("OracleInProcServer.XOraSession")
```

ويمكنك أيضا تكوين دورة عمل لها اسم باستخدام طريقة CreateNamedSession. غير أنه يقوم معظم المبرمجين بوضع التعليمات البرمجية CreateObject في داخل التطبيق الخاص بهم مباشرة. وتكون هذه التقنية أحد الوحدات التي تمثل دورة عمل جاهزة لكي يتم ربطها بقاعدة البيانات.

وكما هو موضح في طرق OraSession في الجدول ١٨-١، فإن وحداتها هي مفتاح العمل مع OO4O. ووظيفتها الأساسية هي تكوين اتصالات قاعدة البيانات بالإضافة إلى أوامر التحكم في المعاملة.

الجدول ١٨-١: طرق OraSession

الطريقة	البيان
BeginTrans	تبدأ هذه الطريقة أحد المعاملات.
CommitTrans	تنفذ هذه الطريقة أحد المعاملات.
ConnectSession	تعيد هذه الطريقة أحد المراجع إلى دورة العمل مع الاسم المورد.
CreateNamedSession	تكون هذه الطريقة دورة عمل باستخدام الاسم المورد.
LastServerErrReset	تعيد هذه الطريقة ضبط معلومات الخطأ الخاصة بالخادم.

الجدول ١٨-١: طرق OraSession

الطريقة	البيان
OpenDatabase	تفتح هذه الطريقة وحدة OraDatabase
ResetTrans	تقوم هذه الطريقة بإلغاء تعديلات كل المعاملات المستحقة.
ResetTrans	تقوم هذه الطريقة بإلغاء إحدى المعاملات.

وبمجرد أن تقوم بتكوين إحدى الدورات سواء كانت دورة افتراضية أو دورة ذات اسم، فإنك الآن من الممكن أن تقوم بعمل اتصالاً مع قاعدة البيانات. ولفتح اتصال مع قاعدة البيانات داخل الدورة، استخدم طريقة OpenDatabase الخاصة بوحدة OraSession. وصياغتها هي كالتالي:

```
Set oradatabase = orasession.OpenDatabase (database_name,
connect_string, options)
```

وتلك هي المعاملات الخاصة بهذه الطريقة:

database_name: ويحدد هذا المعامل اسم الخدمة التي تخص OracleSQL*Net/Net8.

Connect_string: ويحدد هذا المعامل اسم المستخدم وكلمة السر للاتصال بقاعدة البيانات الخاصة بأوراكل.

Options "خيارات": ويضبط هذا المعامل خيارات قاعدة البيانات راجع الجدول (١٨-٢). لاحظ أن 0040 يعتبر الخيارات أطوار للوصول إلى قاعدة البيانات وهو أحد التعبيرات التي تتوافق مع 0040.

الجدول ١٨-٢: خيارات طريقة OpenDatabase

الثابت	القيمة	البيان
ORADB_DEFAULT	صفر	VISUAL BASIC MODE (افتراضي). إذا لم تقم بتخصيص قيمة لأحد الحقول باستخدام طريقة AddNew أو طريقة Edit. فإن القيمة يتم ضبطها

الجدول ١٨-٢: خيارات طريقة OpenDatabase

الثابت	القيمة	البيان
		على NULL بغض النظر عن قيود عمود DEFAULT. كما يقوم هذا الطور بالقفل مع طريقة Edit أو بالانتظار حتى يحدث هذا القفل.
ORADB_ORAMODE	١	Oracle Mode. ويسمح هذا الثابت لقاعدة البيانات بضبط قيم العمود الافتراضية باستخدام AddaNew أو Edit. ثم يقوم بإحضار القيم المستخدمة بالفعل.
ORADB_NOWAIT	٢	Lock No-Wait Mode. ولا يقوم هذا الثابت بالانتظار لأي قفل قد تم تطبيقه من قبل المستخدمين الآخرين حتى يتم رفعه. (يسبب أخطاء ORA-54).
ORADB_NO_REFETCH	٤	Oracle Mode (No Refetch). وهو يتشابه مع Oracle Mode غير أن البيانات لا يتم إعادة إحضارها إلى التخزين المؤقت المحلي بعد إدراج أو تحديث الصفوف.
ORADB_NONBLK	٨	Nonblocking Mode. وليس له أي تأثير على بيئة ٣٢ بت.

تلميح لإمكانية تضمين الثوابت في جدول ١٨-٢ يجب تضمين الملف ORACONST.TXT في المشروع الخاص بك.

وتوضح تعليمات البرمجة ١٨-١ أحد الإجراءات القابلة للتعديل بسهولة حتى يتوافق مع أي معطيات لديك لتوظيف طريقة OpenDatabase من الناحية العملية. وعليك فقط القيام بتغيير اسم خدمة قاعدة البيانات بالإضافة إلى اسم المستخدم وكلمة السر حسب الموقف.

تعليمات برمجة ١٨-١

```
Dim OraSession As Object
Dim OraDatabase As Object
Sub ConnectToDatabase()

' Create the OraSession Object
  Set OraSession = CreateObject("OracleInProcServer.XoraSession")
' Create the OraDatabase Object by opening a connection to Oracle.
  Set OraDatabase = OraSession.OpenDatabase("movies", _
              "video_user/ummagumma", ORADB_DEFAULT)
End Sub
```

متى يمكن أن ترى الخيارات في الجدول ١٨-٢٢؟ يعد الخيار ORADB_ORAMODE من الخيارات المفيدة إذا قمت باستخدام الزنادات لتعيين قيم المفتاح من المسلسل. وفي الواقع فإن هذه هي الطريقة العملية الوحيدة لتجميع المسلسلات مع المجموعات الديناميكية القابلة للتحديث. (راجع "استخدام الزنادات" في هذا الفصل). ومرة أخرى إذا كنت تستخدم مجموعات ديناميكية قابلة للتحديث فإن الخيار ORADB_NOWAIT هو الخيار الوحيد لتجنب قطع التطبيق عندما يواجه أحد الأقفال في أحد الصفوف، وعليك محاولة تصيد أي أخطاء.

ملاحظة يوجد للبرنامج OO40 وحدات OraDynset مشتقة من وحدة Dynaset الأصلية في DAO. دائما ما يستخدم المصطلح المجموعة ديناميكية بوجه عام للإشارة إلى مجموعات النتائج القابلة للتحديث، رغم أن OO40 يستخدم هذا المصطلح لمجموعات البيانات القابلة للقراءة فقط.

غير أنه إذا أردت جعل تطبيقك آمنا وقابلا للتدرج، فإنه من المستحسن أن تستخدم (ORADB_DEFAULT)الافتراضي، وتستخدم SQL أو الإجراءات المخزنة لتحديث أو تعديل البيانات. وفي هذه الحالة فإنك سوف تقوم بتكوين المجموعات الديناميكية باستخدام خيارات ORADYN_NOCACHE و ORADYN_READONLY لمحاكاة مؤشر firehose المتاح في تقنيات ODBC وOLE_DB.

طرق وحدات قاعدة البيانات

سوف تحتاج إلى عدة طرق لاستعادة البيانات من أوراكل بمجرد اتصالك بقاعدة البيانات. وتتضمن قائمة الطرق المتاحة الموضحة في الجدول (١٨-٣) أربع طرق لتكوين OraDYNASET وهي عبارة عن استجابة أوراكل إلى DAO Recordset. وسوف يتم تفصيل أثنين من هذه الطرق في هذا الفصل، أما الطريقتان الأخريان فهما عبارة عن متغيرات تسمح لك بتعديل المعاملات. وتتيح الطرق الأخرى من OraDatabase تنفيذ تعليمات برمجة SQL كما تقوم بتكوين الاستعلامات المخزنة. وكما يتضح فإن وحدة Database هي محور النشاط في OO4O.

الجدول ١٨-٣: طرق OraDatabase.

الطريقة	البيان
Close	تغلق هذه الطريقة قاعدة البيانات.
CreateCustomDynaset	تكون هذه الطريقة أحد المجموعات الديناميكية مع المعاملات الموالفة.
CreateDynaset	تكون هذه الطريقة إحدى المجموعات الديناميكية.
CreatePLSQLDynaset	تكون هذه الطريقة مجموعة ديناميكية PL/SQL (من متغير المؤشر)
CreatePlsqlCustomDynaset	تكون هذه الطريقة مجموعة ديناميكية PL/SQL مع المعاملات الموالفة.
CreateSQL	تكون هذه الطريقة وحدة OraSQLStmt.
ExecuteSQL	تنفذ هذه الطريقة أحد أوامر SQL.
LastServerErrReset	وتعيد هذه الطريقة تحديد آخر خطأ للخادم.

تكوين OraDynaset

يمكن بعد أن قمنا بفتح وحدة قاعدة بيانات للتطبيق أن نقوم بتكوين OraDynaset للاحتفاظ بنتائج الاستعلامات التي تريد أن تنفذها. وتستطيع وحدة OraDynaset أن تحتفظ بالنتائج من أي

استعلام صالح، مثل كل الأفلام الأقصر من ١٢٠ دقيقة. قم بتكوين وحدة OraDynaset باستخدام طريقة CreateDynaset الخاصة بوحدة Database.ومثل كل التقنيات الأخرى للوصول إلى البيانات، تقوم هذه الطريقة بتخليق أد المراجع لوحدة OraDynaset الجديدة، لذلك يجب أن نقوم بتعيينها إلى متغير OraDynaset باستخدام أمر Set.

ملاحظة تتضمن وحداتOLE الخاصة بأوراكل المجموعات الديناميكية، وهي التي يتم مقارنتها بنفس الشيء في DAO. غير أنه لا يوجد ما يساوي هذه اللقطة.

وصيغة طريقة CreateDynaset هي:

```
Dim OraDynaset As Object
Set oraDynaset = oraDatabase.CreateDynaset(source, options)
```

وتشتمل هذه الطريقة على المعاملات التالية:

source "المصدر": وهي عبارة عن مسلسل يمثل مصدر بيانات OraDynaset الجديدة. وسوف تكون من تعليمات SQL البرمجية التقليدية مرفقة في إقتباسين مرفقين. ولا يجب وضع العلامة(;) وهي الفاصلة المنقوطة باللغة الإنجليزية في آخر السطر.

Options "الخيارات": وهي عبارة عن توليفة اختيارية من الثوابت التي تقوم بتحديد صفات OraDynaset الحديثة.ومن الممكن أن تقوم باستخدام واحدا أو أكثر من الثوابت الموضحة في الجدول ١٨-٤.

الجدول ١٨-٤: القيم الممكنة لعلامة خيارات CreateOraDynaset.

الثابت	القيمة	البيان
ORADYN_DEFAULT	صفر	يوضح هذا الثابت السلوك الافتراضي.
ORADYN_NO_AUTOBIND	١	يوقف هذا الثابت عملية ربط المعاملات بالاستعلام. استخدم هذه القيمة إذا كان الاستعلام لا يقوم باستخدام المعاملات.
ORADYN_NO_BLANKSTRIP	٢	يوقف هذا الثابت عملية اقتطاع الفراغات من

```
    ("SELECT movie_id, movie_name FROM video_store.movies", _
      ORADYN_DEFAULT)
  If Not dynMovies.EOF Then

  ' Display the first record.
    MsgBox dynMovies.Fields("movie_id").Value & " : " & _
         DynMovies.Fields("movie_name").Value
    MsgBox "Updatable = " & dynMovies.Updatable
  End If
End Sub
```

لا تحتوي وحدات أوراكل الخاصة بـOLE على أي مشاكل بشأن العناصر الخاصة بـأوراكل مثل أسماء المخططات والعروض البسيطة (جدول واحد) والمترادفات. وفي اللحظة التي تقوم فيها باستخدام المزيد من عناصر SQL المعقدة، مثل بدائل الأعمدة أو ارتباطات الجداول ثم مجموعة نتيجة أوراكل ستكون لحظة للقراءة فقط.

ومن الأشياء الطريفة التي تستطيع عملها هي القيام بتضمين الدوال والعمليات الحسابية في استعلامك. فعلى سبيل المثال، إذا استخدمت:

```
SELECT year_of_release + 2 FROM movies
```

فيمكنك بالفعل الإشارة إلى الحقل

```
dynMovies .Fields (year_Of-realease  + 2")
```

وذلك على الرغم من ضرورة التأكد من الكتابة الإملائية صحيحة تماما حتى آخر المسافة. وهذه الطريقة تسري حيث إن أوراكل يقوم بتسمية بالتسمية الافتراضية لأي عمود معقد بالاسم الذي تم تخصيصه في الاستعلام. وهذا الحقل لن يكون قابلا للتحديث.

وتتمتع المجموعة الديناميكية بمزيد من الحرية التي تستطيع بها السيطرة على متطلباتها للحصول على مجموعات نتائج قابلة للتحديث إذا ما قورنت بطرق ODBC. حيث تتطلب ODBC أن يكون لمجموعات النتائج مفتاح فريد، بينما تقوم الطريقة الخاصة بأوراكل بإضافة معرف الصف قائمة الأعمدة المطلوبة، وذلك إذا كان الأمر متاحا. وبتلك الطريقة، فإنه من الممكن أن تكون مجموعة النتائج قابلة للتحديث دائما حتى مع عدم وجود المفتاح الفريد.

تلميح إذا أردت تعديل الذاكرة الوسيطة واستدعاء المعاملات، يمكنك الاستفادة من طريقة أوراكل المميزة. انظر المقطع"توليف المجموعة الديناميكية لأوراكل" لمزيد من التفاصيل.

الاستفادة من الزنادات

عند استخدامك لمرادف من المجموعة الديناميكية القابلة للتحديث في طرق أخرى من طرق الوصول للبيانات، فسوف تقع في مشكلة ما إذا كنت تستخدم المسلسلات في توليد المفاتيح الأساسية الخاصة بجداولك. والسبب في ذلك أن المفتاح الأساسي لا يمكن أن يكون خاليا (بالتعريف)، ولذلك عند استخدامك طريقة التحديث، يلزمك توفير قيمة لها؛ غير أنه لن يمكنك معرفة ما هي القيمة إذا لم تقم أولا بتنفيذ أمر SELECT آخر للبحث عنه، مثل:

```
SELECT My Table. NEXTVAL FROM dua 1
```

والحل الوحيد لهذه المشكلة هو تمرير قيمة افتراضية في التحديث (مثل الصفر) ثم وضع القيمة الحقيقية في الجدول عن طريق الزناد. وتكمن المشكلة في أن المجموعة الديناميكية سوف يكون لديها صفر أثناء ما يكون العمود المقابل في قاعدة البيانات يحتوي على الرقم الحقيقي المستخدم. ولكن لن يكون بإمكانك تحديث هذا السجل مرة أخرى دون تنشيط المجموعة الديناميكية.

أما OO4O فلديه اتجاه آخر بشأن هذا الأمر حيث تم تصميمه مع وضع مشكلة التسلسل في الاعتبار. فإذا قمت باستخدام الخيار الصحيح (ORADYN_ORAMODE)، عندئذ يمكنك التأكد من أنه في كل مرة تقوم فيها باستخدام طرق AddNew أو Edit، فسوف يتم استرجاع أي قيم افتراضية استخدمتها قاعدة البيانات أو تم تخصيصها عن طريق الزنادات من قاعدة البيانات إلى المجموعة الديناميكية. وتوضح قائمة ١٨-٣ تعليمات البرمجة التي يمكنك كتابتها في دورة SQL*PLUS بهدف تكوين زناد على جدول العملاء. وتقوم تعليمات البرمجة بمراجعة ما إذا كان السجل الجديد جاري إدراجه أم لا. وإذا كان Movie_id الخاص بالسجل الجديد صفرا، عندئذ يتم استبدالها في قيمة من مولد التسلسل.

تعليمات برمجة ١٨-٣

```
CREATE OR REPLACE TRIGGER video_store.customer_insert
 BEFORE
 INSERT
 ON video_store.customers
 FOR EACH ROW
```

```
WHEN (NEW.customer_id = 0)
DECLARE
  v_id  NUMBER(6);

BEGIN
  SELECT CUSTOMER_SEQUENCE_ID.NEXTVAL INTO v_id FROM dual;
  :NEW.customer_id := v_id;
END;
```

وبمجرد تكوينك لزناد، فيمكنك حينئذ كتابة تعليمات البرمجة الخاصة بنا مع العلم أنه في أي مرة تقوم فيها سجل جديد إلى المجموعة الديناميكية باستخدام Movie_id للصفر، فلن يتم استخدام المسلسل الصحيح التالي الخاص بالمفتاح الأساسي فحسب، ولكن سيظهر أيضا الرقم الذي تم استخدامه في المجموعة الديناميكية. وتوضح تعليمات البرمجة الموجودة في القائمة ١٨-٤ هذا التأثير. وقمنا في مثال تعليمات البرمجة باستخدام مربع رسالة لتوضيح أي رقم مسلسل قد تم استخدامه. ولكن لاحظ أنه يجب عليك التأكد من وجودك في السجل الصحيح. وتقوم طريقة AddNew بإضافة سجل جديد على نهاية المجموعة الديناميكية، ولهذا فنحن نقوم بعمل MoveLast لعرض القيمة الصحيحة.

تعليمات برمجة ١٨-٤

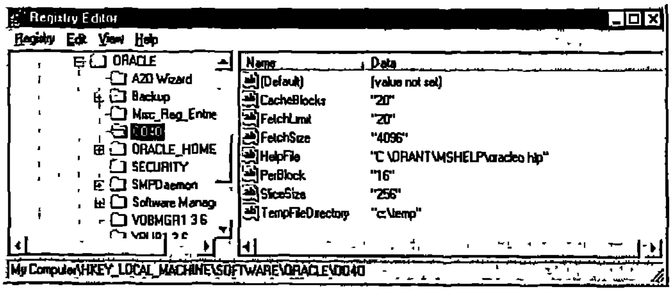
```
Set dynMovies = OraDatabase.CreateDynaset _
    ("SELECT * FROM video_store.customers WHERE last_name = " & 
_
    " :cust_name", ORADYN_ORAMODE)

dynMovies.AddNew
dynMovies.Fields("customer_id") = 0
dynMovies.Fields("first_name") = "Mark"
dynMovies.Fields("last_name") = "Berry"
dynMovies.Fields("drivers_license") = "1243886-12"
dynMovies.Update
dynMovies.MoveLast
MsgBox dynMovies.Fields("customer_id")
```

وتحتاج القيم التي قد تم إحضارها وإنتاجها عن طريق قاعدة البيانات إلى جولة إضافية على شبكة العمل. وعلى كل حال، فإن هذه هي الطريقة التي تعمل بها المجموعة الديناميكية القابلة للتحديث، لذلك تتم الإضافات بشكل بسيط. غير انه إذا أخذ ذلك في الاعتبار فإنه يمكنك أن تستخدم الثابت ORADYN_NO_REFETCH لتجنب العودة إلى الخادم.

توليف المجموعة الديناميكية لأوراكل

تعتبر طريقة OO4O من أفضل طرق الوصول للبيانات التي يمكن توليفها ولكن يعتبر هذا المر من الأشياء التي تسبب ارتباكا للمستخدمين الجدد. وهناك نوعان مختلفان تماما من المعاملات الخاصة بعملية التوليف وهما المعاملان Cache و Fetch. ويمكن إعداد هذه المعاملات في السجلات الموضحة في شكل ١٨-٣.



الشكل ١٨-٣
توليف المجموعة الديناميكية في السجلات.

تعتبر الذاكرة الوسيطة مساحة من الذاكرة التي تقوم بتخزين بيانات المجموعة الديناميكية محليا على جهاز العميل؛ ويوجد ذاكرة وسيطة واحدة لكل مجموعة ديناميكية. ويتم ملء الذاكرة الوسيطة إذا ما تطلب الأمر أن تكون البيانات بيانات مجموعة ديناميكية قابلة للتحديث أو بيانات للقراءة فقط، وبعد ذلك يتم إحضار مجموعة النتائج إلى الذاكرة الوسيطة. ويمكن لهذا المعامل تكوين طلب كبير للذاكرة إذا كنت تقوم بإحضار كميات ضخمة من المعلومات، كما يمكن للمعامل أن يتجاوز بسهولة الحد المتاح للذاكرة. وتقوم طريقة OO4O بتقديم آلية متدفقة عن طريق تحريك البيانات من الذاكرة الوسيطة إلى ملفات البيانات المحلية وذلك عند الاضطرار إلى إحضار بيانات جديدة.

وعلى الرغم من قيام طرق ODBC بعمل قدر معين من التخزين لك، فإن عملية تخزين كل البيانات الموجودة في المجموعة الديناميكية الخاصة بك تعني أنه لديك المزيد من موضوعات إدارة الذاكرة. غير أن طريقة OO4O تعطيك معاملات الذاكرة الوسيطة التي تساعدك في توليف التطبيق الخاص بك. هناك ثلاثة أنواع من معاملات الذاكرة الوسيطة:

SliceSize: القيمة الصغرى لأعداد البايت المستخدمة في تخزين حقل فـــردي. إذا كـــانت جميع حقولك قصيرة، فينبغي إنقاص الوضع الافتراضي الذي يكون ٢٥٦.

PerBlock: عدد الشرائح لكل ذاكرة أو لكل مقطع ملف. وهذا ما يمثل عادة عدد الحقـــول لكل مقطع. ويكون حجم المقطع(SlicSize*PerBlock).

CachBlocks: القيمة العظمى لعدد المقاطع المخزنة في الذاكرة.

وإذا قمت بضرب كل هذه القيم معا ستصل إلى أن (٢٥٦*١٦*٤٠٩٦)=KB٨٠. وهـــذا لا يعني ذاكرة كبيرة قياسا بمعايير اليوم. وبافتراض أنك لا تحتفظ بالمجموعات الديناميكية مفتوحـــة تزامنياً، فلا مانع بالتأكيد من زيادة حجم الذاكرة، خاصةً باستخدام معامل CachBlocks.

ملاحظـــة: يمكنـــك إغـــلاق الذاكـــــرة الوســـيطة كليـــةً باســـتخدام خيـــار ORDYN_NOCACHE. وفـــي هـــذه الحالـــة ســـتحصل علـــى وضـــع للأمام فقط بالنسبة للمؤشـــر.

يتضمن النوع الثاني من أنواع توليف المجموعة الديناميكية معاملات السحب. ويعتبر استخدام هذه المعاملات أسهل من عملية توليف الذاكرة الوسيطة. وهناك نوعان من معاملات السحب:

FetchLimit: عدد الصفوف المقروءة من مجموعة نتائج أوراكل إلى الذاكرة الوسيطة فـــي انتقال واحد. ويجب التأكد أن الذاكرة الوسيطة كبيرة بالقدر الذي يكفي لتخزين حد الســـحب. ويعتبر هذا المعامل بمثابة نقطة تناوب بين إنقاص عدد رحلات الشبكة الإرتجاعية التي تعـود منها مرورا بنقطة البداية وبين زيادة متطلبات الذاكرة الخاصة بهذا التطبيق.

FetchSize: حجم الفاصل الخاص بأعمدة LONG وLONG RAW.

ولا يوجد هناك أي قواعد يسيرة لمجموعة ديناميكية معينة، ولكن في الغالب يكون الإجـــراء الأمثل هو القيام بالمحاولة والخطأ. غير أن الشيء الأكيد هو: عند توليف أي نظام، قم بتغيير فقط في كل مرة، ثم اختبر تأثيره.

تلميح تكون إعدادات السجل الخاصة بتوليـف المعـاملات افتراضيـة بالنسـبة لأي مجموعة ديناميكية لأوراكـل تـم تكوينـها. ولتجـاوز هـذه القيـم يمكنـك اسـتخدام طريقتـي CreateCustomDynaset وCreatePlsqlCustomDynaset. كما يمكنـك اسـتخدام العديـد مـن خصائص المجموعة الديناميكية التي سنراها في المقطـع التـالي، علـى الرغم أنه في هذه الحالة سيكون لأي تغيـيرات تقـوم بـها تـأثير عنـد تنشيط المجموعة الديناميكيـة.

خصائص المجموعة الديناميكية

هناك العديد من الخصائص كما يتضح من جدول ١٨-٥. ويعتبر معظم هذه الخصـائص مـن مرادفات الخصائص بنفس الاسم لـDAO. ولكن الاختلافات الحقيقية هي الخصائص الإضافيـة الستة الخاصة بإعداد الذاكرة الوسيطة وعناصر السحب.

جدول ١٨-٥: خصائص المجموعة الديناميكية لأوراكل.

الخاصية	البيان
BOF	تقوم هذه الخاصية بتعليم بداية الملف.وتكون القيمـة صحيحـة إذا قمت بالتحريك قبل السجل الأول في مجموعة السجلات
Bookmark	تقوم هذه الخاصية بإيقاف المؤشر علـى سـجل فـي مجموعـة السجلات
BookMarkable	تقوم هذه الخاصية بالدلالة على ما إذا كنت تستخدم علامات الكتب
CachBlocks	تقوم هذه الخاصية بإعداد أو إعادة أكبر عدد مـن المقـاطع فـي الذاكرة الوسيطة
CachChanged	تقوم هذه الخاصية بالدلالة على ما إذا كانت المعـاملات قـد تـم تغييرها
CachSliceSize	تقوم هذه الخاصية بإعداد أو إعادة عدد الحجم الخاص بكل شريحة
CachSizePerBlock	تقوم هذه الخاصية بإعداد أو إعادة عدد الشرائح لكل مقطع

جدول ١٨-٥: خصائص المجموعة الديناميكية لأوراكل.

الخاصية	البيان
Connection	تقوم هذه الخاصية بإعادة وحدة ارتباط أوراكل للمجموعة الديناميكية
Database	تقوم هذه الخاصية بإعادة وحدة قاعدة بيانات أوراكل للمجموعة الديناميكية
EditMode	تقوم هذه الخاصية بالدلالة موقف التحرير للسجل الحالي
EOF	تقوم هذه الخاصية بتعليم نهاية الملف. وتكون القيمة صحيحة إذا قمت بالتحريك بعد آخر سجل في مجموعة السجلات
FtchLmiit	تقوم هذه الخاصية بإعداد أو إعادة حجم مصفوفة السحب(عدد الصفوف المسحوبة في انتقال واحد)
FetchSize	تقوم هذه الخاصية بإعداد أو إعادة حجم فاصل مصفوفة السحب
Fields	تمثل هذه الخاصية مجموعة من الحقول المرتبطة بالمجموعة الديناميكية
LastModified	تقوم هذه الخاصية بإعادة علامة الكتاب إلى آخر صف تم تعديله
NOMatch	تقوم هذه الخاصية بالدلالة على انه لم يتم العثور على أي تطابق لطريقة البحث
Options	تقوم هذه الخاصية بإعادة علامة الخيارات المستخدمة لتوليد مجموعة ديناميكية
RecordCount	تحتوي هذه الخاصية على عدد السجلات الموجودة في المجموعة الديناميكية
RowPosition	تحتوي هذه الخاصية على رقم الصف الحالي في المجموعة الديناميكية
Session	تقوم هذه الخاصية بإعادة وحدة دورة أوراكل الخاصة بالمجموعة الديناميكية

جدول ١٨-٥: خصائص المجموعة الديناميكية لأوراكل.

الخاصية	البيان
SQL	تشتمل هذه الخاصية على شريط الاستعلام الخاص بالمجموعة الديناميكية
Tansactions	تدل هذه الخاصية على ما إذا كنت تستطيع الحركات (دائما صحيحة)
Updatable	تدل هذه الخاصية على ما إذا كانت المجموعة الديناميكية قابلة للتحديث

ومن النقاط الجديرة بالملاحظة هي أن خاصية RecordCount تعطيك بالفعل عدد السجلات الإجمال الموجود في المجموعة الديناميكية، حتى إذا استخدمت خيار NOCACHE. وهذا أمر غير معتاد حيث إنه لا يمكن لأي طريقة من الطرق الأخرى التي قمنا بتغطيتها في هذا الكتاب القيام بهذا. وتعتبر خاصية RecordCount الموجودة في طرق ODBC هو عد محلي للسجلات التي قامت بملأ مجموعة السجلات. أمل في حالة OO4O، فيتم اختيار عدد السجلات من خادم أوراكل وسوف ترى هذا الأمر في جزء SQL*Plus.

استكشاف طرق مجموعة OraDynaset

بفرض أنك قمت بفتح مجموعة أوراكل الديناميكية، عندئذ يمكنك استخدام مجموعة مختلفة من الطرق للتحكم في السجلات والأعمدة الموجودة في المجموعة الديناميكية. وتتضمن هذه الطريقة التنقل خلال المجموعة الديناميكية وتغيير قيم الأعمدة والبحث عن العناصر التي تتطابق مع معايير معينة. وتدعم طريقة OO4O مجموعة من الطرق الغنية والواسعة إذا ما قورنت بأساليب الوصول إلى البيانات الأخرى، مع وجود ADO مرتبة ترتيبا جيدا. وترجع هذه الوظيفة الإضافية في OO4O أساسا إلى الارتباط الوثيق مع خادم أوراكل والمرونة التي يتمتع بها عن طريق الاضطرار إلى الاتصال مع RDBMS واحد.

ملء جدول OraDynaset

تعتبر عملية ملء جدول ديناميكي عملية مشابهة لمعظم أساليب الوصول للبيانات الأخرى. يتم سحب البيانات من الخادم في مصفوفة سحب والتي يكون حجمها محددا عن طريق استخدام

خاصية FetchLimit التي تعرضنا في المقطع"توليف المجموعة الديناميكية لأوراكل". وتقوم هذه الخاصية بتحديد أكبر عدد من الصفوف التي تم سحبها إلى العميل في انتقال واحد ومع رحلة قصيرة داخل الشبكة. وهذه البيانات يتم إحضارها إلى الذاكرة الوسيطة. وعند محاولتك للوصول لبيانات غير موجودة في الذاكرة الوسيطة أو في حقول النسخ، عندئذ يجب استدعاء مجموعة الصفوف التالية.

والفرق الوحيد بين مجموعة ديناميكية قابلة للتحديث ومجموعة ديناميكية للقراءة فقط هو أن جملة SELECT التي تستخدمها لتكوين مجموعة ديناميكية قابلة للتحديث قد تم تعديلها لتشتمل على شبه العمود ROWID. فعلى سبيل المثال، إذا كانت الجملة الخاصة بك هي:

```
SELECT movie_id , movie_name rating FROM  video_store.movies
```

فسوف تصبح الآن:

```
SELECT ROWID, movie_id ,movie_name rating FROM
video_store.movies
```

ويمكنك أن ترى في هذا المثال أنه إذا كان لديك ارتباط جدول، فلن يكون لديك ROWID فريد، وبالتالي فلن تكون المجموعة الديناميكية قابلة للتحديث.

ومن البدائل الخاصة بعملية ملء المجموعة الديناميكية هو أنه عند استخدام خيار ORDADYN_NOCACHE يمكنك الحصول على مجموعة ديناميكية للأمام فقط دون ذاكرة وسيطة. كما يمكنك تجنب عملية ملء المجموعة الديناميكية تماما باستخدام خيار ORADYN_NO_MOVEFIRST. وفي هذه الحالة فأنت لا تقوم بإحضار أي صفوف من قاعدة البيانات، ولهذا فهي عملية أسرع إلى حد ما. ويعتبر هذا الخيار من أكثر الخيارات أهمية إذا كنت لا تريد رؤية أي من المعلومات الموجودة وتريد فقط إضافة سجلات جديدة.

تغيير السجل الحالي

عند تكوينك مجموعة ديناميكية لأول مرة، فدائما ما يتم وضعها على السجل الأول(على الرغم أنه يمكن تجنب هذا الأمر عن طريق استخدام الثابت ORADYN_NO_MOVEFIRST في علامة خيارات OraDynaset). هناك ثمانية طرق يتم استخدامها بطريقة مثلى للتحرك حول مجموعة ديناميكية قابلة للتمرير مثل الموضحة في جدول ١٨-٦. كما يمكنك تنفيذ تلك الطرق من تعليمات البرمجة أو أزرار على وسيط رسومي. لاحظ أنه يمكن استخدام العديد من تلك الطرق مع رقم مقاصة من الصفوف.

جدول ١٨-٦: طرق تحريك مجموعة التقارير.

الطريقة	البيان
MoveFirst	تقوم هذه الطريقة بالانتقال إلى السجل الأول في مجموعة السجلات
MoveLast	تقوم هذه الطريقة بالانتقال إلى السجل الخير في مجموعة السجلات
MoveNext	تقوم هذه الطريقة بالانتقال إلى السجل التالي في مجموعة السجلات
MoveNextn	تقوم هذه الطريقة بنقل سجلات المقاصة للأمام، إلى المكان الذي تكون فيه المقاصة هي المعامل الخاص بالطريقة
MovePrevious	تقوم هذه الطريقة بالانتقال إلى السجل السابق في مجموعة السجلات
MovePreviousn	تقوم هذه الطريقة بنقل سجلات المقاصة للخلف
MoveRel	تقوم هذه الطريقة بنقل سجلات المقاصة من الوضع الحالي(من الخلف أو من الأمام)
MoveTo	تقوم هذه الطريقة بالانتقال إلى مقاصة رقم الصف

أما بخصوص DAO، فيجب التحقق من خاصية EOF (نهاية الملف) وذلك عند استخدام طريقة MoveNext كما يجب التحقق من خاصية BOF (بداية الملف) عند استخدام طريقة MovePrevious. أما إذا قمت بمحاولة طرق Move هذه عندما يكون كل من EOF وBOF على وضع True، فربما ينتهي بك الأمر بسجل حالي غير صحيح. أما إذا كانت مجموعة التقارير فارغة، فسوف يكون كل من BOF EOF على وضع True.

الوصول إلى حقول المجموعة الديناميكية لأوراكل

للوصول إلى حقول مجموعة ديناميكية للقراءة فقط أو قابلة للتحديث من المجموعات الديناميكية لأوراكل، يمكنك اتباع واحدة من بين طريقتين مختلفتين. وتعتبر الطريقة الأولى مشابهة لطريقة DAO التي تقوم فيها باستخدام مجموعة حقول المجموعة الديناميكية. وللبحث عن القيمة الخاصة بأي حقل من المجموعة الديناميكية الحالية، يمكنك الرجوع للحقل حسب الفهرس (عدد ترتيبي) أو حسب الاسم. أما عند استخدام طريقة OO4O، فلابد من اسم الخاصية "Value". وفيما يلي التركيب الخاص بالوصول إلى الحقل:

```
OraDynaset.Fields("move_id").Value
```

أو

```
OraDynase("move_id").Value
```

وكانت نسخة 0040 التي سبقت نسخة 2.2 معروف عنها أنها بطيئة في عملية استرجاع قيم الحقل بتلك الطريقة وذلك بسبب الوصول المتكرر إلى مجموعة الحقول. ويجب عند استخدام أي نسخة من 0040 استخدام الطريقة الثانية المتاحة وإعداد وحدة حقل لكل حقل تريد استرجاعه كما هو موضح في قائمة ١٨-٥.

تعليمات برمجة ١٨-٥

```
Dim fld1 As Object
Dim fld2 As Object
Set fld1 = OraDynaset.Fields(0)
Set fld2 = OraDynaset.Fields(1)
'  Now use the temporary objects
MsgBox fld1.value
MsgBox fld2.value
```

بالطبع هناك بعض المصاريف الإضافية عند تكوين هذه الوحدات الإضافية، وسوف تكون في حاجة لاستخدام هذه الطريقة فقط عند تنقلك خلال عدد كبير من السجلات. أما إذا كان لديك حقول عديدة، فيجب أن تقوم بإظهار مصفوفة من وحدات الحقل بدلا من إظهار كل وحدة على حدة.

```
Dim flds(10) As Object
Set fld(0) = OraDynaset.Fields(0)
Set fld(1) = OraDynaset.Fields(1)
```

وللقيام بتعبئة مربع القائمة المعتاد، يمكنك استخدام تعليمات البرمجة الموضحة في قائمة ١٨-٦. وكما ترى في القائمة، فمربع القائمة يستخدم مجموعة ديناميكية للقراءة فقط وللأمام فقط(دون وجود ذاكرة وسيطة) حيث إننا نريد تجنب المشاكل التي قد يسببها مؤشر التمرير القابل للتحديث.

تعليمات برمجة ١٨-٦

```
Public Sub FillListBox(ListBox As Control)
'  Fill a list box with movie names
```

```
' Ignore error trapping
  Dim rs As Object
  Dim fld As Object
  Set rs = OraDatabase.CreateDynaset _
     ("SELECT movie_name FROM video_store.movies", _
      ORADYN_READONLY + ORADYN_NOCACHE)
  Set fld = rs.Fields(0)

  ListBox.Clear
  While Not rs.EOF
     ListBox.AddItem fld.Value
     rs.MoveNext
  Wend
  ListBox.ListIndex = 0 ' Display the first item
End Sub
```

تغيير السجلات والقيم الموجودة في مجموعة ديناميكية

هناك عدد قليل من الطرق التي يمكننا استخدامها للقيام بتغيير السجلات وقيمها الموجودة في مجموعة ديناميكية بافتراض انك تقوم باستخدام مجموعة ديناميكية قابلة للتحديث. أولا، يجب الانتقال إلى السجل الذي تريد تغييره باستخدام طرق Move ثم تقوم باستخدام طرق التحرير الآتية:

AddNew: تقوم هذه الطريقة بإضافة سجل جديد إلى المجموعة الديناميكية ولكن ليس إلى قاعدة بيانات أوراكل. ولجعل التغيير دائم، يجب استخدام طريقة التحديث.

Edit: تقوم هذه الطريقة بوضع السجل الحالي في طور التحرير بمعنى أنه يمكنك تغيير أي قيمة من القيم.

Update: تقوم هذه الطريقة بحفظ أي تغييرات قمت بإجرائها على قاعدة بيانات أوراكل، بافتراض انه تم موافاة كل القيود.

Delete: تقوم هذه الطريقة بحذف السجل الحالي. لا يمكنك استخدام طريقة التحديث بعد القيام بالحذف.

وإذا قمت بتنفيذ طريقتي AddNew وEdit، فيمكنك القيام بتغيير أي الحقول الموجودة في المجموعة الديناميكية لأوراكل. وفيما يل مثال على هذه الطريقة:

```
OraDynaset("movie_id").Value = "The Wonderful World of Oracle"
```

لا تحتوي طريقة OO4O على أي مراجعة وذلك عند تخصيص قيم جديدة للحقول (مقارنة بطريقة JET التي بعملية مراجعة غير متقدمة لنوع الحقل). وسوف تجد أن عملية مراجعة الحقل قد تم الانتهاء منها عن طريق أوراكل وذلك عند استخدامك طرق Update، ولذلك فإنه من الممارسة الجيدة قيامك بالتأكد من عمل أكبر قدر من المراجعة باستخدام فيجوال بيسك بقدر ما تستطيع. أما إذا كان مشاكل بمراجعة البيانات أو مشاكل في القيود مثل مفتاح غير صحيح أو إذا كانت هناك قيمة حرفية بينما تريد قيمة رقمية، عندئذ سوف تحصل على خطأ يمكن الوقوع فيه مرة أخرى.

وعند استخدامك لطريقة Edit، فإن OO4O يقوم بوضع قفل في الصف المتأثر باستخدام أمرSEKLECT FOR UPDATE دائم. أما إذا كان لمستخدم آخر قفل بارز على نفس الصف (تغيير غير مثبت)، عندئذ سيتم تجميد التطبيق. وبالطبع سوف يستغرق هذا وقتا طويلا إذا لم تقم بكتابة تطبيقك واضعا هذا الأمر في الاعتبار. وللأسف فمثل هذه الأقفال تتيح احتمالية وضع مستخدم لقفل على صف من صفوف قاعدة البيانات ثم الانصراف عنه لفترة طويلة.

تلميح بدلا من ترك تطبيقك يتجمد عند محاولته الوصول إلى سجل مغلق، يجب استخدام خيار ORADB_NOWAIT من طريقة تكوين قاعدة بيانات ومنع أي أخطاء ناتجة.

عند قيامك بإضافة سجل جديد إلى المجموعة الديناميكية، يجب أن تتأكد أن كل الحقول المطلوبة (تلك الحقول التي لديها قيد NOT NULL) ذات قيمة صحيحة. وهذا الأمر يمكن أن يسبب مشكلة، خاصةً إذا كنت تستخدم مسلسلات للمفتاح الأساسي. أما إذا كنت تقوم باستخدام مجموعات ديناميكية قابلة للتحديث، عندئذ يجب قراءة الجزء" الاستفادة من الزنادات" الذي سبق شرحه في هذا الفصل. أما إذا التزمت بالمجموعات الديناميكية الخاصة بالقراءة فقط، فيمكنك استخدام الإجراءات المخزنة أو تنفيذ تعليمات البرمجة SQL للقيام بتحديث البيانات ثم تنشيط المجموعة الديناميكية. وبعد تنفيذ هذا الأمر، ستحصل على نظام شامل أفضل.

البحث عن سجل معّين

الشيء الأخير الذي سنبحث عنه في طريق المجموعة الديناميكية هو كيفية البحث عن سجل يطابق المعايير التي قمت بإرسالها. ومن الغريب أن طريقة OO4O تدعم بيئة البحث الغنية هذه، في الوقت الذي لا يقوم خادم أوراكل بتدعيم أمر FIND وكما أن طرق وصول البيانات مثل RDO وADO قامت بإسقاط البحث نهائياً. تم تأسيس طريقة OO4O على DAO/JET التي قامت

باستخدام محرك JET لتقوم بكل معالجات FIND. وفي OO4O يتم معالجة أمـــر Find محليـــا، ومن الممكن أن يتم استخدام أي صيغة في جملة Find التي يمكن استخدامها في جملـــة Where من عبارة select ويتضمن هذا:

- عبارات شرطية عـــادي مثـــل "Movie_Name = 'Awakenings'".
- وظـــائف Oracle مثـــل "LENGTH(Movie_Name) <100".
- الاستعلام الفرعـــي.

وهناك أربعة طرق تخص Find وخاصية واحدة لوظيفة Find ويتم عرض كـــل هـــذا فـــي الجدول ١٨-٧. مع ملاحظة أنهم جميعا متوافقين تماما مع DAO.

الجدول ١٨-٧: الطرق والخصائص المستخدمة في سجلات Find.

الطريقة/الخاصية	البيان
FindFirst	وتعثر هذه الطريقة على أول حدوث لسلسلة البحث في المجموعـــة الديناميكية.
FindLast	وتعثر هذه الطريقة على آخر حدوث لسلسة البحث في المجموعـــة الديناميكية.
FindNext	وتعثر هذه الطريقة على الحدوث التالي لسلسة البحث بعد الســـجل الحالي.
FindPrevious	وتعثر هذه الطريقة على الحدوث السابق لسلسلة البحث قبل الســـجل الحالي.
NoMatch	وتشير هذه الخاصية إلى أنه لم يتم العثور علـــى مطـــابق لأمـــر FIND.

وكما هو الحال مع DAO، فإنه يجب استخدام هذه الأوامر على نحو بسيط لأنها دليل علـــى أنك قد قمت بأخذ المزيد من البيانات أكثر من حاجتك من قاعدة البيانات. ومن الممكن أن يمـــون هذا مقبولا إذا كنت تمتلك مجموعة نتائج للقراءة فقط وتتميز بأنها ثابتة إلى حد ما. وإلا فإنه يتـــم وضع جملة Find في جملة Where في عبارة Select الابتدائية الخاصة بـــك.وإذا كنـــت قـــد

قررت استخدام طريقة Find فإن تعليمات البرمجة ١٨-٧ توضح كيفية عملها.ويجب ملاحظة أنه بعد إتمام المطابقة الناجحة من الممكن تنفيذ الأمر Command مع غير جملة Find.

تعليمات برمجة ١٨-٧

```
Dim strSearchClause As String

strSearchClause = "LENGTH(movie_name) < 100 "
rsMovies.FindFirst strSearchClause
If rsMovies.NoMatch Then
   MsgBox "Couldn't find movie "
Else
   MsgBox "Movie ID for 'Titanic' is " & rsMovies("movie_id").Value
End If
```

ملاحظة يستخدم OO4O محلل محلي لتفسير جمل Find والذي من شأنه أن يوفر رحلة إضافية في الخادم. ويمكن في الإصدارات الحديثة من OO4O (على الأقل حتى النسخة ٢-٣) تجنب أية جملة تحتوي على اقتباس مفرد مضمن، وإلا فإن محلل سوف يعطي رسالة خطأ.

إغلاق المجموعة الديناميكية

طريقة Close، لإغلاق عنصر Recordset. وتعد طريقة Close هي الطريقة الوحيدة التي يتم توفيرها في OO4O ولذلك يمكن ترجمة التعليمات البرمجية من DAO إلى OO4O ببساطة وفي الواقع، لا تقوم طريقة Close بعمل أي شيء، حيث يتم تدمير المجموعة الديناميكية آلياً بمجرد خروجها خارج المجال.

ستخدام المعاملات

يجب مراجعة مجموعة OraParameter أولاً قبل الدخول إلى أية موضوعات أخرى، ورغم أنه يمكن كتابة تطبيقات كاملة بدون الالتفات إلى المعاملات، إلا أنه لن يصبح في الإمكان التشغيل أعلى من المستوى DAO/JET. ويعطي استخدام بطريقة OO4O وظائف أكثر من أي طريقة

أخرى للدخول إلى البيانات. ويتم إعداد مجموعة OraParameters واحدة لكل عنصر Database، بينما هناك مجموعة Parameter واحدة لكل rdoQuery في RDO.

وكما رأيت في الفصل ٩ "مقدمة عن PL/SQL"، يسمح أوراكل بإعداد المتغيرات التي يمكن استخدامها لمتغيرات الإدخال والإخراج بالنسبة للإجراءات المخزنة. وعند تكوين متغير، يمكن استخدامه في عبارة SQL عن طريق وضع نقطتان قبله، فمثلا بدلا من وضع السطر التالي.

```
DELETE FROM  movies WHERE movie_id = 9;
```

يمكن إعداد معرف متغير مترابط وتعيين قيمة له ثم كتابة:

```
DELETE FROM  movies WHERE movie_id = :id;
```

ويقوم أوراكل بتغيير قيمة المتغيرات المترابطة إلى عبارة SQL بمرور الوقت ولكن الميزة في استخدام هذا النوع من المتغيرات في أن أوراكل يقوم بتقليل العبارة مرة واحدة فقط، بينما تستطيع مرات التنفيذ المتتابعة الاستفادة من Shared Pool ثم ربط قيم المتغيرات الجديدة ببساطة بدون إعادة تحليل العبارة وبذلك تصبح المتغيرات المترابطة أكثر سرعة عندما تكون SQL على الأقل بالتنفيذ الأول.

ملاحظة تمثل معاملات SQL المترابطة معاملات IN فقط من الناحية الفنية. وعند استخدام معاملات وضع IN OUT، فإنه يكون هناك متغيرات مترابطة PL/SQL. ويعتني Oracle Server بربط المتغيرات.

يتيح OO4O القيام بنفس العملية تماما مثل ما يحدث عند استخدام SQL أولا، يلزم إنشاء معاملات مجموعة OraParameters بطريقة Add كما يلي:

```
OraDatabase.OraParameters.add Name, InitialValue, Type
```

يحتوي OO4O على العبارات التالية:

Name: وتحدد اسم ملزم غير أنه مميز للمعامل والذي يجب توفيره من جانب المبرمج.

InitialValue: وتحدد القيمة التي يقوم المبرمج بإدخالها للمعامل في أول مرة.

Type: وتحدد وضع المعامل، ويتم إيضاح الإمكانيات في الجدول ١٨-٨.

الجدول ١٨-٨: ثوابت النوع المتاحة لمجموعة ORAPARAMETERS.

الثابت	القيمة	وضع PL/SQL
ORAPARM_INPUT	1	IN
ORAPARM_OUTPUT	2	OUT
ORAPARM_BOTH	3	IN OUT

وهناك شيء يجب عمله قبل استخدام المعاملات وهو توفير نوع الخادم لكل معامل، حيث يقوم نوع الخادم بتحديد نوع البيانات لكل معامل مترابط. ولتحديد النوع يجب استخدام خاصية server type الخاصة بعنصر parameter بالقيم الموضحة في الجدول ١٨-٩. ويلزم اتخاذ الحذر للتأكد من الحصول على البيانات الصحيحة التي لم يتم هذا فإنه يمكن أن يواجه المبرمج بعض الصعوبة عند محاولته اكتشاف سبب توقف المعامل البديل عن العمل.

الجدول ١٨-٩: الثوابت المتاحة لخاصية server type.

الثابت	القيمة	نوع البيانات الخارجية
ORATYPE_VARCHAR2	1	VARCHAR2
ORATYPE_NUMBER	2	NUMBER
ORATYPE_SINT	3	SIGNED INTEGER
ORATYPE_FLOAT	4	FLOAT
ORATYPE_STRING	5	Null Terminated STRING
ORATYPE_VARCHAR	9	VARCHAR
ORATYPE_DATE	12	DATE
ORATYPE_UINT	68	UNSIGNED INTEGER
ORATYPE_CHAR	96	CHAR
ORATYPE_CHARZ	97	Null Terminated CHAR
ORATYPE_CURSOR	102	PL/SQL Cursor

ورغم أن الاستخدام الأساسي لهذه المعاملات يكون مع استعلامات الإجراء المخزن وعنصر OraSQLStmt، فإنه يمكن أيضا استخدام معاملات الإدخال مع أي عبارة SQL. ويتم عرض مثاليين في تعليمات البرمجة ١٨-٨ عن كيفية القيام بذلك وتقوم التعليمات البرمجية بإضافة معامل جديد Cust_id له قيمة افتراضية ٦ ثم تقوم بضبط نوع البيانات ويعرض المثال الثاني نفس العملية غير أنه في هذه الحالة يقوم بضبط معامل سلسلة حرفي Cust_name وتكون قيمته الافتراضية Berry.

تعليمات برمجة ١٨-٨

```
' Customer_id parameter
  OraDatabase.Parameters.Add "cust_id", 6, ORAPARM_INPUT
  OraDatabase.Parameters("cust_id").ServerType =
ORATYPE_NUMBER
  Set dynMovies = OraDatabase.CreateDynaset _
    ("SELECT * FROM video_store.customers " & _
     " WHERE customer_id = :cust_id", ORADYN_ORAMODE)

' Customer_name parameter
  OraDatabase.Parameters.Add "cust_name", "Berry",
ORAPARM_INPUT
  OraDatabase.Parameters("cust_name").ServerType =
ORATYPE_VARCHAR
  Set dynMovies = OraDatabase.CreateDynaset _
    ("SELECT * FROM video_store.customers " & _
     " WHERE last_name = :cust_name", ORADYN_DEFAULT)
```

وفي هذه الحالة التي لا تتسم بالتعقيد يمكن أيضا عدم توفير خاصية ServerType لأنه يمكن العمل بشكل بدون الحاجة إليها وإذا كانت هناك حاجة إلى إعداد استعلامات لها معاملات مثل التي تم تعريفها سابقا فيجب مراجعة القسم "استخدام OraSQLStmts" لأنه يتعرض إلى طريقة مفيدة لاستخدام استعلامات SELECT.

وبعد تنفيذ أحد الأوامر التي تستخدم المعاملات يمكن تغيير قيم معامل أو أكثر ثم إعادة تنفيذ أحد الإجراءات المخزنة أو إعادة إنشاء مجموعة ديناميكية. ويمكن ضبط أو استعادة قيم للمعاملات عن طريق الإشارة إلى خاصية Value كما يلي:

OraDatabase.Parameters("cust_name").Value = "Gilmour"

وعند الانتهاء من أحد المعاملات، يجب حذفه من مجموعة OraParameters بطريقة Remove.

ملاحظة إذا تمت استعادة متغيرات إخراج CHAR أو VARCHAR أطول من ١٢٨ بايت، فإنه يلزم ضبط خاصية MinimumSize للمعامل على قيمة كبيرة حتى تستطيع احتواء هذه القيمة.

تحذير لا يستطيع OO4O التنبؤ بالمعاملات التي سوف يتم ربطها بأحد استعلامات SQL أو الإجراء المخزن وذلك لأن مجموعة parameter مرتبطة بعنصر Database ويقوم أوراكل افتراضياً بتجريب كل معامل في المجموعة لمعرفة إمكانية ربطه من عدمها. وإذا كان هنا عدد كبير من المعاملات. فإن هذه الطريقة. يمكن أن ينتج عنها بعض المشاكل رغم أن هذا يحدث فقط عند تنفيذ تعليمات برمجية SQL لأول مرة. ولتجنب ربط المتغيرات غير المطلوبة، يتم استخدام خيار ORADYN_NO_AUTOBIND عند إنشاء المجموعة الديناميكية dynaset بدون معاملات أو عن طريقة إبطال عمل كل المعاملات غير المرغوبة باستخدام طريقة AutoBindDisable.

تنفيذ أوامر SQL

لتحقيق مقدار أوفر من الاتزان والأمن فإنه سوف يتم الابتعاد عن مجموعات النتيجة القابلة للتحديث والاقتراب من تنفيذ أوامر SQL ثم الإجراءات المخزنة. ويحتاج المبرمج إلى طريقة جديدة فيما يخص أوامر DML مثل INSERT وUPDATE وDELETE ولا يختلف OO4O كثيراً عن الطرق الأخرى للوصول إلى البيانات في أن لها طريقة Execute رغم أنها تسمى في هذه الحالة ExecuteSQL.

وأحد خصائص أمر ExecuteSQL هو إمكانية استعادة عدد الصفوف المتأثرة. ومن المعروف أنه من غير الممكن استعادة هذا العدد بالطرق المبنية على ODBC وصيغة ExecuteSQL هي:

```
rowcount = oradatabase.ExecuteSQL (sql_statement)
```

حيث sql_statement هو أمر صالح يخص SQL. ويتضمن ذلك مجموعات الأوامر واستعدادات الإجراءات المخزنة الموجودة في عبارات BEGIN وEND لكتل PL/SQL غير المعروفة ويتم عرض أحد الأمثلة على استخدام هذا الأمر في تعليمات برمجة ١٨-٩.

تعليمات برمجة ١٨-٩

```
Dim iRowCount As Integer

IRowCount = OraDatabase.ExecuteSQL( _
   "DELETE FROM customers WHERE customer_id < 100")
MsgBox iRowCount & " customers deleted!"
```

تحذير تقوم طريقة ExecuteSQL بتنفيذ أمر COMMIT آليا، وهو الأمر الذي قد يكون غير متوقع، ورغم أن ذلك قد يبدو مفيدا، إلا أن ذلك قد يؤثر على أية معاملات يتم استخدامها. (عند العمل على عنصر OraSession، فإن ذلك يؤثر على كل شيء) ولكن لحسن الحظ، فإنه يمكن تضمين طريقة ExecuteSQL في أحد المعاملات وهو الأمر الذي يجب عمله حيث يؤدي إلى تحسن ملحوظ في الأداء.

استدعاء الإجراءات المخزنة

ليس هناك فارق كبير بين تنفيذ عبارة DML القياسية مثل INSERT أو UPDATE أو DELETE وتنفيذ أحد الإجراءات المخزنة. ويمكن تنفيذ الإجراء المخزن عن طريق الوصول إلى البيانات الأخرى، فعلى سبيل المثال، فإن التعليمات البرمجية الموضحة في المثال التالي تقوم بنفس الوظيفة التي تقوم بها مع DAO.

تعليمات برمجة ١٨-١٠

```
Private Sub MovieInsert(iMovieId As Integer, strMovieName As String
Ilength As Integer, strRating As String, iYear As Integer)
'  Insert a new movie into the database
Dim strSQL As String
```

```
StrSQL = "BEGIN video_store.movies_insert2 ("
StrSQL = strSQL & iMovieId & ", '" & strMovieName & "', "
StrSQL = strSQL & iLength & ", '" & strRating & "', " & iYear
StrSQL = strSQL & "); END;"
OraDatabase.ExecuteSQL strSQL
End Sub
```

عبر أن ذلك يقيد المبرمج ويجعله منحصراً في معاملات الإدخال فقط في حين أنه ملزم ببناء الاستعلام في كل مرة. فإذا كان لديك أحد الإجراءات يحتوي على ١٥ معامل، فإن هـذا يعنـي مزيداً من العمل الإضافي وبدلاً من هذا المنهج الصعب، فإنه يمكـن اسـتخدام معـاملات لقيـم الإدخال واستعادة أية قيم للإخراج أولاً دعنا بعد إجراء فيجـوال بيسـك لإنشـاء أي معـاملات مطلوب في هذه الوحدة، وهو كالآتي:

تعليمات برمجة ١٨-١١

```
Private Sub CreateParameters()
' Create the database parameters
' Notice that this works without defining ServerType
  OraDatabase.Parameters.Add "movie_id", 0, ORAPARM_BOTH
  OraDatabase.Parameters.Add "movie_name", "", ORAPARM_INPUT
  OraDatabase.Parameters.Add "length", 0, ORAPARM_INPUT
  OraDatabase.Parameters.Add "rating", "", ORAPARM_INPUT
  OraDatabase.Parameters.Add "year", 0, ORAPARM_INPUT
End Sub
```

والآن دعنا نستخدم هذا المعامل في أحد الإجراءات لإدراج البيانات فـي قـاعدة البيانـات. وتعرض تعليمات برمجية ١٨-١٢ كيفية حدوث ذلك. ويجب ملاحظة أنه لا يتم تخليق عنصـر استعلام كما تم ذلك مع rdoQuery، غير أنه يتم إعادة تخليق العبارة في كل مرة. ويمكن حـل هذه المشكلة مع عنصر OraSQLStmt، وهو ما يتم شرحه فيما بعد.

تعليمات برمجة ١٨-١٢

```
Private Sub MovieInsert(iMovieId As Integer, _
        StrMovieName As String,_iLength As Integer, _
```

```
        StrRating As String, iYear As Integer)
' Insert a new movie into the database

    OraDatabase.Parameters("movie_id").Value = 0
    OraDatabase.Parameters("movie_name").Value = strMovieName
    OraDatabase.Parameters("length").Value = iLength
    OraDatabase.Parameters("rating").Value = strRating
    OraDatabase.Parameters("year").Value = iYear

    OraDatabase.ExecuteSQL "BEGIN movies_insert3 (:movie_id, " & _
            ":movie_name, :length, :rating, :year); END;"

    MsgBox "The movie_id used was " & _
        OraDatabase.Parameters("movie_id").Value
End Sub
```

ملاحظة: من أحد مميزات OO4O عند تنفيذ الإجراءات المخزنة هو إمكانية استخدام المتغيرات المترابطة في أي وقت في أحد الاستعلامات. بينما في ODBC وOLE-DB يجب أن يتم استخدام عنصر Parameter واحد في كل معامل رسمي في الإجراءات المخزنة.

استعادة متغيرات المؤشر

تم التعرض لمتغيرات المؤشر في الفصل ١٠ "تكوين الإجراءات المخزنة"، وتعد أحد الطرق المعقدة لاستفادة البيانات من قاعدة بيانات أوراكل وأحد العناصر التي تؤكد كذب المفهوم الذي يقر بأن أوراكل يستطيع إعادة مجموعة نتيجة من أحد الإجراءات المخزنة. ففي الواقع يستطيع أوراكل القيام بمثل هذه المهمة، غير أنها عملية ليست مبسطة بالإضافة إلى أنها ليست متاحة في طرق، ODBC فضلا عند كونها غير متاحة حاليا عبر OLE-DB.

ومتغير المؤشر هو أحد المؤشرات PL/SQL مفتوح على الخادم الذي يتم استعادته من الإجراء المخزن. ويظل المؤشر مفتوحا على الخادم حتى يتم انتهاء التعامل معه. وهو مؤشر للقراءة فقط وللأمام فقط شأنه كشأن كل مجموعات نتيجة أوراكل. وتكون القدرة على تمرير جانب المزود متاحة عبر OO4O، غير أنها تظل للقراءة فقط.

وللوصول إلى متغير المؤشر، يجب استخدام طريقة خاصة وهي طريقة CreatePLSQLDynaset، والتي لها البنية التالية:

```
Set OraDynaset = CreatePLSQLDynaset(SQLStatement, CursorName,
Options)
```

وتحتوي القائمة التالية على عبارات هذه الطريقة

SQLStatement: وتحتوي على الإجراء المخزن PL/SQL الخاص ببرنامج أوراكل أو أي قطعة تعليمات برمجية PL/SQL.

CursorName: وتحتوي على اسم المؤشر الذي تم إنشاؤه في الإجراء المخزن PL/SQL.

Options: وتحتوي على علامة يشير إلى موقف أية حالات اختيارية من المجموعة الديناميكية. ويمكن جمع الخيارات عن طريق إضافتها. والثوابت المتاحة التي يمكن استخدامها يتم شرحها في الجدول ١٨-١٠، وتمثل مجموعة فرعية من الخيارات لطريقة CreateDynaset.

الجدول ١٨-١٠: ثوابت الخيارات لطريقة.

الثابت	القيمة	البيان
ORADYN_DEFAULT	صفر	ويقبل هذا الثابت السلوك الافتراضي.
ORADYN_NO_BLANKSTRIP	٢	ويمنع هذا الثابت الفراغات المتتابعة في بيانات مسلسل الحرف من اقتطاعها.
ORADYN_NOCACHE	٨	ويمنع هذا الثابت إنشاء بيانات لمجموعة ديناميكية محلية. وهذا المؤشر بكفاءة أكثر غير أنه للأمام فقط.
ORADYN_NO_MOVEFIRST	٦٤	ويتجنب هذا المؤشر طريقة MoveFirst على المجموعة الديناميكيةBOF وEOF يكون TRUE. ولهذا الثابت قيمة بسيطة لأننا نقوم بإنشاء المجموعة الديناميكية للقراءة فقط.

وفيما يلي يتم عرض الأمثلة المسبقة التي تتيح استعادة مجموعة نتيجة مسبقة على الاستعلام التالي

```
SELECT * FROM movies WHERE year_of_release = 1995
```

وكما رأينا مسبقا، يمكن التعامل مع قيمة year_of_release على أساس متغير مترابط [:year]، فيما يؤدي بنا في النهاية إلى النتيجة التالية:

```
SELECT * FROM movies WHERE rating = :year
```

ويلزم إنشاء أحد الإجراءات المخزنة أولا حتى يمكن استعادة متغير المؤشر المبني على هذا الاستعلام. وقد تم إنشاء هذا الإجراء للمخزن في الفصل العاشر تعليمات برمجية ١٠-٨. والآن يمكن تخليق برنامج OO40 حتى يتسنى قراءة متغير المؤشر فيه. ويجب تذكر أن الإجراءات المخزنية يجب أن يتم إنشاءها في حزمة برمجية حتى يتسنى استعادة مجموعة النتائج راجع الفصل ١٠ لتذكر بكيفية القيام بهذه العملية.

وفي الحزمة البرمجية movie_package يوجد الإجراء التالي:

```
PROCEDURE get_old_movies (
  p_year            NUMBER,
  old_movies IN OUT movie_cv_type);
```

وبعد اكتمال الإجراء المخزن يجب إعداد طريقة CreatePLSQLDynaset في التطبيق. وتوضح التعليمات البرمجية ١٨-١٣. ويجب ملاحظة أنه يتم إنشاء معامل واحد فقط للاستعلام، ويتم الإعلان ضمنيا عن المتغير المترابط الآخر OldMovies في التعليمات البرمجية CreatePLSQLDynaset كما أنها تتحول آليا إلى المجموعة الديناميكية.

تعليمات برمجة ١٨-١٣

```
Dim OraSession As Object
Dim OraDatabase As Object
Private Sub GetOldMovies(iYear As Integer)
   Dim OraDynaset As Object

' Create the year_of_release parameter
   OraDatabase.Parameters.Add "year", 0, ORAPARM_INPUT
   OraDatabase.Parameters("year").ServerType = ORATYPE_NUMBER

   OraDatabase.Parameters("year").Value = iYear
 ' Create OraDynaset based on the "OldMovies" cursor
   Set OraDynaset = OraDatabase.CreatePLSQLDynaset _
    ("Begin movie_package.get_old_movies(:year, :OldMovies); end;", _
```

```
    "OldMovies", 0&)
   If Not OraDynaset.EOF Then
 ' Display the first record.
      MsgBox OraDynaset.Fields("movie_id").Value & " : " & _
          OraDynaset.Fields("movie_name").Value
   End If
End Sub
```

سوف تكون الإجراءات المكتوبة التي تستخدم متغيرات المؤشر في غاية البساطة. ولا يمكن بناء مجموعة نتائج خاصة بالمبرمج، غير أنها تكون الناتج من عبارة SELECT. وفي الحالة التي تم شرحها آنفاً، ثم تمرير أحد المتغيرات المستخدمة في جملة WHERE الخاصة بعبارة SELECT. ويمكن من الخاصية العملية إضافة أي شيء إلى جملة WHERE سواء كانت على المستخدم أو يوم من الأسبوع، أو أي شيء أخر. ويعد استخدام متغيرات المؤشر أكثر مرونة من استخدام العرض، غير أن كل منهما له نفس ميزة إخفاء الجدول الأساسي عن المستخدم.

تلميح إذا كانت هناك رغبة في تعديل الذاكرة الوسيطة مع استعادة المعاملات، فمن الممكن الاستفادة من طريقة CreatPlsqlCustomDynaset.

استدعاء وظائف PL/SQL

دائماً ما يتم إهمال استدعاء وظائف PL/SQL وفي الواقع فإن استدعاء وظائف PL/SQL سهل مثل استدعاء الإجراءات المخزنة غير أنه يجب أن تكون واعياً لنقطتين في. أولاً: يجب أن تقوم بإعداد معامل إخراج حتى يتسنى التحكم في قيمة المردود، وثانياً: يجب استخدام صيغة تعيينات PL/SQL إلى المتغير المرتبط.ويتم توضيح هذا في تعليمات برمجة ١٨-١٤ ويقوم هذا الإجراء الخاص ببرنامج الفيجوال بيسك باستدعاء وظيفة get_movie_id باستخدام معامل إدخال واحد. ويتم عرض وظيفة PL/SQL في تعليمات برمجة ١٥-٦.

تعليمات برمجة ١٨-١٤

```
Private Function GetMovieId(strMovieName As String) As Long
 ' Find the ID for the supplied movie name
 ' First set up the parameters
```

```
OraDatabase.Parameters.Add "movie_name", 0, ORAPARM_INPUT
OraDatabase.Parameters.Add "movie_id", 0, ORAPARM_OUTPUT
Now set the value and execute the function
OraDatabase.Parameters("movie_name") = strMovieName
OraDatabase.ExecuteSQL "BEGIN :movie_id := nick.get_movie_id " &

                "(:movie_name); END;"
Finally print out the return value
MsgBox "Movie ID is " & OraDatabase.Parameters("movie_id").Value
GetMovieId = OraDatabase.Parameters("movie_id").Value
End Function
```

استعادة جداول PL/SQL

وبالإضافة إلى متغيرات المؤشر التي تم إيضاحها فإن الفصل العاشر يقدم جداول PL/SQL وسوف يتم استعادتها على اعتبار أنها تحصيلات لها تأثير في PL/SQL ويتم تمثيلها على شكل مصفوفة تعتمد قيمتها على الصفر(أي أنها تبدأ بالعنصر صفر) في 0040.وعلى أي حال فإن هذه هي الطريقة المعتادة للعمل مع PL/SQL، لذلك فإنها لن تسبب أية مشاكل.

ويجب أولا البحث عن معامل مختلف يمكن إضافته إلى تحصيل OraParameter قبل استخدام جداول PL/SQL. وتسمى هذه الطريقة الجديدة Parameter Array أو مصفوفة المعامل. ويتم إضافة المصفوفة إلى التحصيل باستخدام طريقة AddTable بدلا من طريقة Add.التي تم استخدامها للمعاملات العادية. وصيغة طريقة AddMethod هي:

```
OraDatabase.Parameters.AddTable Name, Type, ServerType,
                ArraySize, ElementSize
```

وتحتوي هذه الطريقة على المعاملات التالية:

Name ويقدم هذا المعامل اسم ملزم ومميز لمصفوفة المعامل التي يجب عليك توفيرها.

Type ويحدد هذا المعامل الطور الذي يمثله هذا المعامل. والاحتمالات هي نفس الاحتمالات الموضحة في الجدول ١٨-٨.

ServerType ويحدد هذا المعامل نوع البيانات لعناصر مصفوفة المعامل التي يتم ربطها بقاعدة البيانات.(راجع الجدول ١٨-٨).

ArraySize ويحدد هذا المعامل عدد العناصر في مصفوفة المعامل بحد أقصى للحد الفاصل ٣٢ كيلو. وتستخدم OO4O هذا العدد لتحديد متطلبات الذاكرة مع الإجراءات المخزنة كحد لحجم الجدول.

Element Size ويحدد هذا المعامل الحجم الأقصى لحقول الحرف.التي يتم استعادتها في الجدول(تترك المعامل لعدد لا حصر له من البيانات).

ولجعل هذه الطريقة في حيز التنفيذ، فسوف يتم تنفيذ أمثلة مختلفة عن الأمثلة الموجودة في الفصل ١٠، وهي تعليمات البرمجة ١٠-٧.وتصميم الإجراء المراد تنفيذه هو:

```
PROCEDURE get_stars (
  p_table_size        NUMBER,
  p_stars        OUT char_table_type)
```

ويوجد هذا الإجراء في الحزمة البرمجية *movie_package*. وكما هو واضح فإنه من المتوقع أن يتم استقبال أكبر حجم من الجدول في المعامل الأول، ويستخدم هذا الحجم لوضع حد لعدد قيم المردود عندما يقوم الإجراء المخزن ببناء جدول PL/SQL. ولاستعادة الجدول *p_stars* يتم إضافة مصفوفة معاملات إلى تحصيل المعاملات، ويتم تنفيذ هذه الخطوة في الجزء الأول من تعليمات برمجة ١٨-١٥.

تعليمات برمجة ١٨-١٥

```
Private Sub GetStars(iNumStars)
   Dim iStar As Integer
 ' First set up the parameters
   OraDatabase.Parameters.AddTable "GetStars", ORAPARM_OUTPUT, _
                         ORATYPE_VARCHAR2, iNumStars, 30
   OraDatabase.Parameters.Add "NumStars", 0, ORAPARM_INPUT

 ' Set the maximum table size required and retrieve the table
   OraDatabase.Parameters("NumStars").Value = iNumStars
   OraDatabase.ExecuteSQL "BEGIN movie_package.get_stars " & _
                       "(:NumStars, :GetStars); END;"

 ' Now we will print out the results
   For iStar = 0 To iNumStars - 1
```

```
    Debug.Print OraDatabase.Parameters("GetStars").get_Value(iStar)
  Next
End Sub
```

ويجب ملاحظة أنه في تعليمات برمجة ١٨-١٥ تم تحديد أن الجدول في معامل طـور OUT. ومن الممكن استعادة القيم من الجدول في فيجوال بيسك عن طريق استخدام خاصية Get_Value بمجرد تنفيذ الإجراء المخزن. وإذا تم تصميم الإجراء المخزن على أساس أن جـــدول PL/SQL كان في طور IN أو IN OUT فإن ذلك يتيح تمرير البيانات إلى الإجراء عن طريـــق اســتخدام خاصية Put_Value.

وإحدى الإمكانيات المفيدة هي رد أحد الجداول الذي يحتوي على نوع معقد من البيانات بـــدلا من النوع البسيط الذي تلتزم به. وسوف يمكننا هذا من استعادة بنية Record مرة أخرى والتـــي سوف تكون أقوى من متغير المؤشر، لأنه من الممكن أن يتم بناء محتوياتها حقـــل بعـــد حقـــل، وصف بعد صف. ورغم أن هذه الإمكانية سوف تكون مفيدة جدا، إلا أنه لا يبدو أن هذه المـــيزة سوف تظهر قريبا. ومن الممكن أن تكون جداول PL/SQL أداة مفيدة ويتم استخدامها من حيـــن لآخر بواسطة PL/SQL المتضمنة. كما أنها تستحق الإضافة إلى المخزن الخاص بك.

استخدام OraSQLStmts

يتيح OO4O الفرصة لتكوين أحد العناصر التي تقوم بتحديد تعليمات SQL كما هو الحـــال مـــع rdoQueries وQueryDefs في الفصل السابق. وتكمن الميزة في هذه التقنية في أنه يتم تحليلها عن طريق خادم أوراكل مرة واحدة ثم يتم استخدامها بعد ذلك عدة مرات. واستخدام هذا العنصر سوف يكون أكثر أهمية من تكوين تعليمات SQL حديثة في كل مرة. وفيما يلي طريقة عمل هـــذه الخطوة باستخدام طريقة SQL:

```
Set orasqlstmt = oradatabase.CreateSQL(sql_statement, options)
```

ومعاملات هذه الطريقة هي:

sql_statement وتقوم بتحديد تعليمات SQL الصالحـــة كمـــا أنـــها ليســـت محـــدودة باستعلامات SELECT.

Options وتقوم بوصف الخيار المراد استخدامه، ومن الممكن أن تقوم باختيار خيار مـــن الخيارات الموجودة في الجدول ١٨-١١ أو واحد منها فقط.

الجدول ١٨-١١: القيم الممكنة لتعليمات الخيارات في طريقة CreateSQL.

الثابت	القيمة	البيان
ORASQL_NO_AUTOBIND	١	ويقوم هذا الثابت بتعطيل ربط المعامل، ويتم استخدام هذا الخيار إذا لم تحتوي التعليمات على معاملات لأنها سوف تكون كافية أكثر.
ORASQL_FAILEXEC	٢	ويقوم هذا الثابت برفع الخطأ كما أنه لا يقوم بتكوين عناصر تعليمات SQL إذا قامت التعليمات برفع الخطأ.

ولاستخدام عنصر OraSQLStmt يجب تكوين مسلسل SQL كما هو موضح في تعليمات برمجة ١٨-١٦. ومن الممكن أن تحتوي تعليمات البرمجة SQL على عدد من المتغيرات المترابطة وتكون واحدة لكل معامل في هذه الحالة. ثم يتم تعيين قيم للمعاملات وتجديد OraSQLStmt ويفضل أن تتم هذه الخطوة في جزء منفصل من البرنامج. وتقوم طريقة التجديد هذه بتنفيذ تعليمات SQL، كما تقوم بربط آخر قيم المعامل.

تعليمات برمجة ١٨-١٦

```
Private ostMovieInsert As Object
' Set up an OraSQLStmt for the insert
  strSQL = "BEGIN movies_insert3 (:movie_id, " & _
        ":movie_name, :length, :rating, :year); END;"
  Set ostMovieInsert = OraDatabase.CreateSQL(strSQL, 0)

' Assign values to the parameters and refresh the OraSQLStmt
  OraDatabase.Parameters("movie_id").Value = 0
  OraDatabase.Parameters("movie_name").Value = "Movie Name"
  OraDatabase.Parameters("length").Value = 122
  OraDatabase.Parameters("rating").Value = "G"
  OraDatabase.Parameters("year").Value = 1998

  OstMovieInsert.Refresh
```

تحديث المجموعة

من أحد الاستخدامات المعروفة لمصفوفات المعامل هي ما يسمى على نحو خاطئ بتحديث المجموعة. وقد تم شرح كيفية ربط المعاملات إلى عبارة INSERT مثلاً على النحو التالي:

```
iRowCount = OraDatabase.ExecuteSQL( _
   "INSERT INTO movies VALUES (:movie_id, :movie_name, " & _
   ":length, :rating, :year, USER, SYSDATE) "
```

وإذا كانت هناك حاجة لتنفيذ ذلك عدة مرات على أحد الصفوف فإنك سوف تحتاج إلى بعض الإضافات الخاصة بالذهاب والعودة لكل صف تريد إدراجه. ومن أحد المشاكل أيضاً في OO4O هي أنك سوف تمتلك مصروفات إضافية أكثر من الطرق الأخرى مثل RDO وذلك لأن التحكم يكون ActiveX. ومن الممكن أن يكون الأداء غير جيد بدون أحد العناصر مثل rdoQuery. غير أن OO4O لديها الحل، وهي أنك لا تستطيع فقط أن تستخدم المعاملات المترابطة في تعليمات INSERT بل مصفوفات المعامل. ومن الممكن باستخدام هذه الطريقة أن تقوم بإعداد أحد المصفوفات من ١٠ إلى ١٠٠ صف ليتم إدراجها وتنفيذها كلها باستدعاء واحد. وسوف تلاحظ التقدم في الأداء.

وسوف يتم إعداد بعض التعليمات البرمجية لإدخال البيانات إلى جدول Customer كمثال على إدراج المجموعة. وسوف نقوم بإدخال ٥٠٠ صف ثم يتم مقارنة التقنيات لاستخدام المدرجات الفردية. وتوضح تعليمات البرمجة ١٨-١٧ التعليمات البرمجية التي تقوم بتنفيذ كل من هذه المهام والأوقات التي يتم تحديدها لذلك. وتتكون التعليمات البرمجية من عدة أجزاء. أولاً يتم إعداد اتصال قاعدة البيانات، ثم يتم تكوين OraSQLStmt للإدراج المتتابع والمعاملات المتعلقة به. ثم يتم تشغيل العبارة ٥٠٠ مرة قبل تجديد OraSQLStmt. ويتكون النصف الآخر من التعليمات البرمجية من تكوين Parameter Array ثم يتم تشغيل الاختبار الذي يقوم بملأ مصفوفات المعامل بعدد ٥٠٠ تسجيل(يجب تذكر أنها مبنية على الرقم صفر)، ثم يقوم في النهاية بتنفيذ عبارة INSERT.

تعليمات برمجة ١٨-١٧

```
Private OraSession As Object
Private oraDatabase As Object
Private Sub Form_Load()
' Test repeated inserts against batch inserts
   Dim StartTime, EndTime
```

```
Dim i As Integer
Dim strSQL As String
Dim ostInsert As Object
Dim OraPArray1 As Object
Dim OraPArray2 As Object
Dim OraPArray3 As Object
Dim OraPArray4 As Object

' Connect to the database
Set OraSession = CreateObject("OracleInProcServer.XOraSession")
Set oraDatabase = OraSession.OpenDatabase("beq-local", _
            "video_user/ummagumma", 0)
' Set up an OraSQLStmt and associated parameters
strSQL = "INSERT INTO customers VALUES(:id, :first_name, " & _
   ":last_name, :license)"
Set ostInsert = oraDatabase.createsql(strSQL, 0)

oraDatabase.Parameters.Add "id", 0, ORAPARM_INPUT
oraDatabase.Parameters.Add "first_name", 0, ORAPARM_INPUT
oraDatabase.Parameters.Add "last_name", 0, ORAPARM_INPUT
oraDatabase.Parameters.Add "license", 0, ORAPARM_INPUT

oraDatabase.executesql ("DELETE FROM  customers")
Start the first test
StartTime = Timer
For i = 1 To 500
   oraDatabase.Parameters("id") = i
   oraDatabase.Parameters("first_name") = "Peter"
   oraDatabase.Parameters("last_name") = "Richards"
   oraDatabase.Parameters("license") = "123456ABCDEF"
   ostInsert.Refresh
Next
EndTime = Timer
```

```
MsgBox "Normal inserts " & EndTime – StartTime & " seconds"


Set up the parameter arrays
oraDatabase.Parameters.AddTable "ids", ORAPARM_INPUT, _
  ORATYPE_NUMBER, 500
oraDatabase.Parameters.AddTable "first_names", ORAPARM_INPUT,
_
  ORATYPE_VARCHAR2, 500, 25
oraDatabase.Parameters.AddTable "last_names", ORAPARM_INPUT,
_
  ORATYPE_VARCHAR2, 500, 25
oraDatabase.Parameters.AddTable "licenses", ORAPARM_INPUT, _
  ORATYPE_VARCHAR2, 500, 20

Set OraPArray1 = oraDatabase.Parameters("ids")
Set OraPArray2 = oraDatabase.Parameters("first_names")
Set OraPArray3 = oraDatabase.Parameters("last_names")
Set OraPArray4 = oraDatabase.Parameters("licenses")


oraDatabase.executesql ("DELETE FROM  customers")

Start the second test
StartTime = Timer
For i = 0 To 499
   OraPArray1.put_Value i + 1, I
   OraPArray2.put_Value "Stanley", i
   OraPArray3.put_Value "Matthews", i
   OraPArray4.put_Value "123456ABCDEF", i
Next
OraDatabase.executesql ("INSERT INTO customers VALUES" & _
   "(:ids, :first_names, :last_names, :licenses)")
EndTime = Timer
```

```
MsgBox "Normal inserts " & EndTime – StartTime & " seconds"

End Sub
```

قد تلاحظ أن هذا الاختبار لا يتناسب مع الحالة الثانية تماماً لأننا كان يجب أن نقوم بإزالة أو تعطيل المجموعة الأولى من المعاملات. غير أن هذا الاختبار يوضح الإدراج المتتابع يحتاج إلى ٨,٨٧٥ ثانية، بينما يستغرق إدراج المجموعة ٠,٨٧١ ثانية. والفائدة من ذلك كانت على جهاز واحدة حيث لا تهم كثيراً سرعة الشبكة.وعلى الرغم من أن rdoQuery مثلاً تعد طريقة كافية لأداء هذه المهمة، إلا أنها لا تقارن بسرعة إدراج المجموعة. غير أن إدراج المجموعة لا يتناسب مع كل المواقف، ولكن إذا واتتك الفرصة، فلا تتردد عن استخدامه.

معالجة الأخطاء

تختلف إمكانيات معالجة الأخطاء الخاصة ببرنامج OO4O عن الإمكانيات الخاصة بالطرق الأخرى للوصول إلى البيانات. ويعمل OO4O على هيئة خادم جاري تشغيله وهو شديد الحساسية لأخطاء التنفيذ الآلي وخاصة الخطأ المؤثر رقم ٤٤٠ وهو "OLE Automation Error". وليس هناك تحصيلError تستطيع معرفة الخطأ الذي حدث منه لذلك فإنك تحتاج إلى فحص خاصية LastServer Error الخاصة بالدورة أو قاعدة البيانات لتجد الخطأ مرفوع بواسطة الخادم الجاري تشغيله. ولاستعادة رسالة نصية عن الخطأ، يتم استخدام خاصية LastServerErrText الخاصة بقاعدة البيانات أو الدورة.

ومن أحد المتغيرات الشيقة الخاصة بالبرنامج OO4O هو أنه يقوم بتمرير موقع مسلسل SQL إلى تطبيق العملي في نفس المكان الذي تم فيه تصيد الخطأ. وربما أنك قد لاحظت هذه الميزة في SQL*Plus فإذا قمت بكتابة أمرSQL صالح فإن SQL*Plus سوف تقوم بالإشارة إلى موقع حدوث الخطأ على المسلسل بالإضافة إلى عرض رسالة الخطأ. وهذه هي إحدى الطرق المفيدة لمعالجة أخطاء النص، غير أنها لا تستخدم كثيراً في بيئة الإنتاج. وإذا كنت تريد استخدام هذه الخاصية، يجب عليك أن تقوم بفحص خاصية LastServerErrPos.

والميزة الأخيرة الخاصة بمعالجة الأخطاء هي إعادة تحديد الخطأ مع طريقة LastServerErrReset.

التجميع

يتم في هذا الجزء اعتبار التطبيق البسيط السابق ذكره غير أنه في هذه المرة يتم التعامل معه مع الأخذ في الاعتبار مميزات OO40. ويستخدم هذا التطبيق العديد من المميزات التي تمت مناقشتها في هذا الفصل. ومن أحد الإجراءات الشيقة CreateParameters (راجع تعليمات برمجة ١٨-١٨) وهو الذي يقوم بإعداد المعاملات لإجراء MovieInsert.ثم تقوم بإعداد مجموعة ديناميكية لها أحد المعاملات لاستعادة المشاركين في الفيلم الحالي. ويتم ضبط هذا المعامل في كل مرة يتم فيها عرض أحد الأفلام على النموذج كما يتم تجديد المجموعة الديناميكية حتى يتم استخدام المعامل الجديد لإنتاج السجلات. وفي النهاية يتم تكوين OraSQLStmt لمعالجة المدرجات التي تم إعدادها مسبقاً.

تعليمات برمجة ١٨-١٨

```
Private Sub CreateParameters()
'  Create the database parameters
'  Notice that this works without defining ServerType
   Dim strSQL As String

   OraDatabase.Parameters.Add "movie_id", 0, ORAPARM_BOTH
   OraDatabase.Parameters.Add "movie_name", "", ORAPARM_INPUT
   OraDatabase.Parameters.Add "length", 0, ORAPARM_INPUT
   OraDatabase.Parameters.Add "rating", "", ORAPARM_INPUT
   OraDatabase.Parameters.Add "year", 0, ORAPARM_INPUT
   OraDatabase.Parameters.Add "find_movie_id", 7, ORAPARM_INPUT


'  Set up a forward-only dynaset to show participants

   strSQL = "SELECT first_name || ' ' || last_name full_name,"
   strSQL = strSQL & " DECODE (role, 'A', 'Actor', 'D', 'Director') "
   strSQL = strSQL & " role, movie_id"
   strSQL = strSQL & " FROM participants, movie_participants"
```

```
    strSQL = strSQL & " WHERE movie_id = :find_movie_id "

    strSQL = strSQL & "  AND movie_participants.participant_id = "
    strSQL = strSQL & " participants.participant_id"
    strSQL = strSQL & "  ORDER BY role"
    Set rsParticipants = OraDatabase.CreateDynaset _
                (strSQL, ORADYN_READONLY + ORADYN_NOCACHE)
'   Set up an OraSQLStmt for the insert
    strSQL = "BEGIN movies_insert3 (:movie_id, " & _
          ":movie_name, :length, :rating, :year); END;"
    Set ostMovieInsert = OraDatabase.CreateSQL(strSQL, 0)
End Sub
```

ويتم عرض تعليمات البرمجة الخاصة بالإجراء cmdAction_Click وهي التي تستجيب إلى رغبة المستخدم في إضافة أو تحديث أو تعديل أحد السجلات.و تقوم طريقة Add في معالجة استدعاء الإجراء المخزن باستخدام OraSQLStmt، على الرغم من أن وظائف Delete و Update تستخدم استدعاءات ذات تعليمات برمجية دائمة. ومن الناحية العملية فإنك سوف تحتاج إلى إعداد أحد عناصر OraSQLStmt لكل إجراء مخزن واستدعاء وظيفي قد تم استخدامه في التطبيق الخاص بك.

تعليمات برمجة ١٨-١٩

```
Private Sub cmdAction_Click(Index As Integer)
'   Execute the required action - insert, update or delete.
    Dim iField As Integer
    Dim strSQL As String
    Dim iRowCount As Integer

    If Index = 0 Then            ' Add a new record
       ShowNewRecord
       strMode = "Add"

    ElseIf Index = 1 Then         ' Update the record
```

```
        If strMode = "Add" Then
            OraDatabase.Parameters("movie_id").Value = 0
            OraDatabase.Parameters("movie_name").Value = txtMovie(0)
            OraDatabase.Parameters("length").Value = CInt(txtMovie(1))
            OraDatabase.Parameters("rating").Value = txtMovie(2)
            OraDatabase.Parameters("year").Value = CInt(txtMovie(3))
            ostMovieInsert.Refresh
            RefreshRecordset

        Else
        '   Use a conventional inline call here
            strSQL = "BEGIN movies_update ("
            strSQL = strSQL & rsMovies("movie_id").Value & ","
            strSQL = strSQL & "'" & txtMovie(0) & "', " ' Movie Name
            strSQL = strSQL & txtMovie(1) & ", "       ' Length
            strSQL = strSQL & "'" & txtMovie(2) & "', " ' Rating
            strSQL = strSQL & txtMovie(3)              ' Year
            strSQL = strSQL & "); END;"
            iRowCount = OraDatabase.ExecuteSQL(strSQL)
        End If
        strMode = ""
        RefreshRecordset

    ElseIf Index = 2 Then          ' Delete the record
        strSQL = "BEGIN movies_delete (" & rsMovies(4).Value & ");
END;"
        iRowCount = OraDatabase.ExecuteSQL(strSQL)
        RefreshRecordset
    End If

End Sub
```

عرض بنية قاعدة البيانات

يتيح OO40 أحد التحصيلات التي تقوم بتفصيل كل الحقول في المجموعة الديناميكية. ويعد تكوين المجموعة الديناميكية هو الطريقة الوحيدة لرؤية البنية. وذلك يختلف عن طرق ODBC التي تتيح لك رؤية أي جدول تستطيع الوصول إليه.ويتم استخدام تحصيلات OraFields في OO40 التي تحتوي على سلاسل من عناصر OraField. ولكل من هذه العناصر مجموعة من الخواص التي يتم عرضها في الجدول ١٨-١٢. ولا توجد هناك طريقة أخرى لتغيير بنية الجدول كما هو الحال مع كل طرق الوصول إلى البيانات فيما عدا طريقة JET. وحتى مع طريقة JET فإنها لا ينصح بها، لذلك فإن إلغاء هذه الميزة لا يشكل مشكلة.

جدول ١٨-١٢: خصائص وحدات OraField.

خاصية **OraField**	البيان
OralDataType	تستعيد هذه الخاصية التعليمات البرمجية الخاصة بنوع البيانات الداخلية الخاص بأوراكل
OraMaxDSize	تستعيد هذه الخاصية الحجم الأقصى للعرض
OraMaxSize	تستعيد هذه الخاصية الحجم الأقصى للعمود غير أنها تعتمد على نوع البيانات.
OraNullOK	تشير هذه الخاصية إلى إذا ما كان قد تم السماح لعناصر NULLs في هذا العمود من عدمه.
OraPrcision	تستعيد هذه الخاصية العدد الأقصى للأرقام في الحقل الرقمي.
OraScale	تستعيد هذه الخاصية عدد الأماكن العشرية في الحقل الرقمي.
Name	تستعيد هذه الخاصية اسم الحقل.
Size	تستعيد هذه الخاصية حجم الحقل بالأرقام أو بالبايت (صفر لحقول LONG).
Truncated	وتشير هذه الخاصية إلى ما إذا كان عمود LONG أو LONG RAW قد تم استرجاع جزء منه.

جدول ١٨-١٢: خصائص وحدات OraField.

خاصية **OraField**	البيان
Type	تستعيد هذه الخاصية التعليمات البرمجية التي تشير إلى نوع بيانات الفيجوال بيسك المتوقعة.
Value	تستعيد هذه الخاصية قيم حقل معين للصف الحالي.

وستحتاج إلى معلومات OraFields إذا كنت تقوم بكتابة أحد التطبيقات العامة والتي تتعدد ردود أفعالها تبعا لنوع البيانات التي تتعامل معها مع افتراض أنك لا تعرف أي الجداول سوف يتم التعامل معها. ويمكن اعتبار برنامج الإدارة متعددة الجوانب مثال على هذه الحالة مثل الذي تم عرضه في الفصل ١٤.

جدول ١٨-١٣: تعليمات البرمجة الخاصة بنوع بيانات أوراكل الداخلية.

نوع بيانات أوراكل	الثابت	نوع البيانات
CHAR	ORADB_TEXT	مسلسل
DATE	ORADB_DATE	متباين
LONG	ORADB_MEMO	مسلسل
LONG RAW	ORADB_LONGBINARY	مسلسل
NUMBER (1-4,0)	ORADB_INTEGER	صحيح
NUMBER (5-9,0)	ORADB_LONG	عدد صحيح طويل
NUMBER (10-15,0)	ORADB_DOUBLE	مزدوج
NUMBER (16-38,0)	ORADB_TEXT	مسلسل
NUMBER (1-15,n)	ORADB_DOUBLE	مزدوج
NUMBER (16-38,n)	ORADB_TEXT	مسلسل

جدول ١٨-١٣: تعليمات البرمجة الخاصة بنوع بيانات أوراكل الداخلية.

نوع بيانات أوراكل	الثابت	نوع البيانات
RAW	ORADB_LONGBINARY	مسلسل
VARCHAR2	ORADB_TEXT	مسلسل

عيوب OO4O

على الرغم من كل المميزات التي تقدمها هذه الطريقة وكل هذه الوظائف التي توفرها والتي قد تعجز عن تقديمها العديد من الطرق الأخرى، إلا أن لها بعض العيوب.

- في بعض الأحوال يتم الأداء بشكل بطيء، ويعزى ذلك جزئيا إلى أنه خادم جاري تشغيله. وحتى إذا قمت باستخدام التقنيات الموجود في ملف OO4O HELP لمحاولة تحسين الأداء، فإن OO4O لن تستطيع أن تجاري DAO/OLE-DB.
- حتى الآن لا يوجد دعم للعمليات اللاتزامنية أو الأحداث.
- لا يوجد دعم لطريقة GetRows حيث أن OO4O قد تم تصميمه للمجموعات الديناميكية القابلة للتحديث.
- تقوم طريقة OO4O بربطك بأوراكل. وإذا ما قررت التغيير إلى قاعدة بيانات أخرى فإن الانتقال إلى RDO أو إلى ADO ليس هينا.
- نادرا ما يأتي ذكر OO4O في كتب قواعد البيانات.

خلاصة

تم تغطية آخر طرق الوصول إلى البيانات التي يرد ذكرها في هذا الكتاب في هذا الفصل؛ "وحدات OLE الخاصة بأوراكل". وقد تم عرض التكامل الوثيق بينها وبين Oracle Server وتوضيح تدعيمها لكل التقنيات التي يدعمها أوراكل لإدخال وإخراج البيانات. ويشتمل هذا على إعداد وصلات قاعدة البيانات وتكوين استعلامات SELECT SQL لاستعادة مجموعة متنوعة من البيانات. وقد تم عرض كيفية استخدام معاملات SQL في استعلاماتنا ومعاملات PL/SQL في الإجراءات المخزنة. كما تم تغطية كيفية تنفيذ أوامر SQL بالإضافة إلى كيفية استعادة معاملات الإخراج ومتغيرات المؤشر وجداول PL/SQL من الإجراء المخزن. كما تعرضنا أيضا لكيفية استخدام الاستعلام المخزن OraSQLStmt، وأخيرا كيفية تصيد الأخطاء في OO4O.

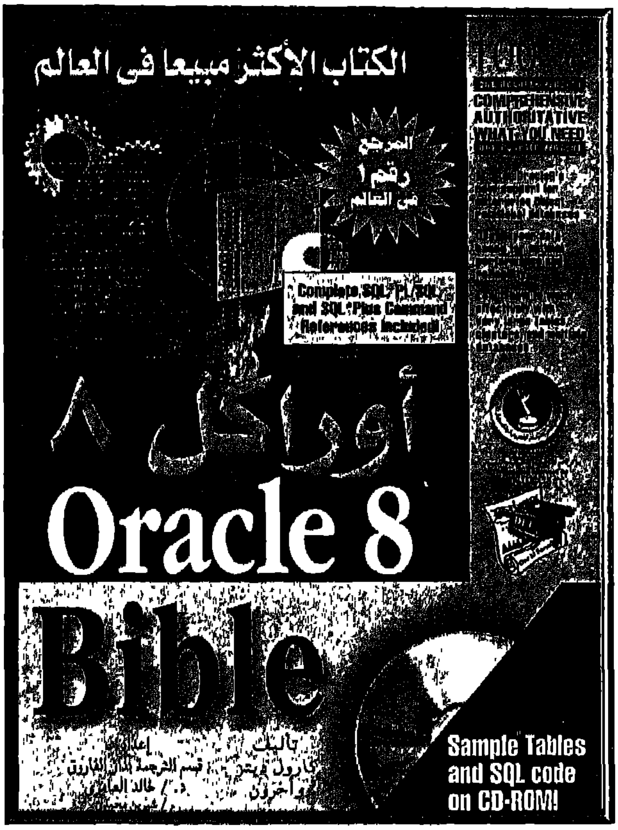
الخاتمة

يحوي هذا الكتاب بين صفحاته قدراً غير قليل من المعلومات، ومما لاشك فيه أنـه كلمـا زادت معرفتك بأوراكل، كلما وجدت التطبيقات الخاصة بالفيجوال بيسك أكثر تشويقاً ونفعاً. وبالمثل فإنه كلما تعلمت أكثر عن تقنيات الوصول للبيانات الخاصة ببرنـامج فيجـوال بيسـك، كلمـا زادت الإمكانات المتاحة لك عند استخدام برنامج أوراكل.

وقد أتاح الإصدار السادس من فيجوال بيسك وADO 2.0 أو OO4O للمـبرمجين إمكانـات جديدة للتعامل مع أوراكل. والتقنيات التي قد قمنا بعرضها في هذا الكتاب تمكنك مـن إصـدار تطبيقات متميزة تستطيع أن تنافس بها التطبيقات المتداولة فـي السـوق. وعلـى الرغـم مـن المحاولات التي تجري لاستكمال الفيجوال بيسك بمحرك Java، فإنه ليس هناك من شكٍ فـي أن مستقبل الفيجوال بيسك مع أوراكل سيكون عظيماً سواء فيما يخص تطبيقات المزود/الخادم أو في بناء طبقات المزود والطبقات الوسطى للتطبيقات متعددة الطبقات.

ومما لاشك فيه أن التقنيات التي قد تم عرضها في هذا الكتاب ما زالت قابلة للتغيـير، فقـد تتغير بعض التفاصيل الخاصة بها، الأمر الذي قد يؤدي إلى توقف التطبيق عن العمل، لذلك فإنـه يجب على القارئ أن يتابع النشرات الدورية أو قواعد البيانات الخاصة بالدعم الفنـي أو مواقـع الويب التي تتناول مثل هذه الموضوعات عن كثب، حيث أن العلم لا يقف عند حد معين.

الآن بالأسواق



أوراكل ٨ بايبل
تأليف ماك كولج دايتر
إعداد قسم الترجمة بدار الفاروق
د/ خالد العامري - م/ شيرين محمدى

Oracle Programming with Visual Basic

The Indispensable Resource for Visual Basic Developers Working in an Oracle Environment

Program for Client-Side or Middle-Tier Applications

Take Full Advantage of Oracle's VB-Friendly Features

Implement Oracle Successfully and Meet Your Company's Data Needs—Large or Small

Technically Reviewed and Approved by

Bonus! Online Web Resources
www.sybex.com

Get the most out of the world's most powerful database— using the world's most popular development tool.

Oracle Programming with Visual Basic is a one-of-a-kind guide to Visual Basic development in an Oracle environment. Inside, you'll learn why Visual Basic is the ideal tool for connecting to Oracle databases; what the Oracle server has to offer VB developers; and how to implement Oracle using Visual Basic's powerful data-access functionality.

This book has everything you need to know about Oracle and Visual Basic to set up your database and deliver the access your organization needs.

Building on your knowledge of Visual Basic, this book helps you master all of its database-connectivity features. With Oracle, it takes you from the ground up, moving from a review of concepts and architecture to in-depth instruction in the techniques—both standard and advanced—you'll use to carry out and fine-tune your implementation. Coverage includes:

- Understanding Oracle concepts and architecture
- Setting up Oracle and creating your database
- Mastering the new Oracle features of Visual Basic 6
- Addressing database-management issues
- Addressing data-modeling issues
- Running SQL queries and updates
- Creating stored procedures with PL/SQL
- Setting up clients for accessing Oracle
- Understanding client-side and middle-tier technology
- Using data-access technology: design issues, visual tools, Data Access Objects
- Using advanced Oracle techniques: Remote Data Objects, ODBCDirect, ActiveX Data Objects, Oracle Objects for OLE
- And much more

Nick Snowdon is a senior software specialist for Westech Information Systems in Vancouver, British Columbia. His twenty years of experience as an engineer includes mathematical modeling and database design. Snowdon is a frequent contributor to Visual Basic Programmer's Journal.

الوكيل الوحيد على مستوى الشرق الأوسط

دار الفاروق للنشر والتوزيع
DAR EL FAROUK

USER LEVEL
BOOK TYPE
CATEGORY

ISBN 0 7821 2322 8
90000
9 780782 123227

دار الفاروق
بوابة القرن الحادى و العشرين